

FIG. 1

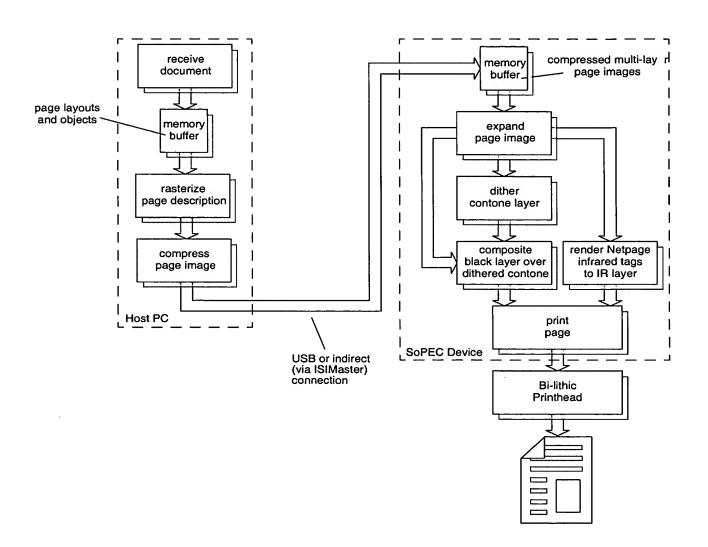


FIG. 2

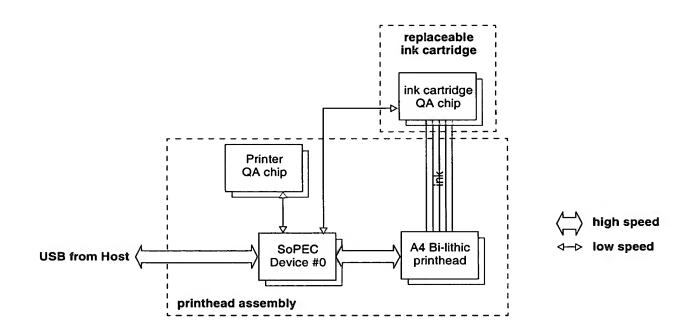


FIG. 3

high speed

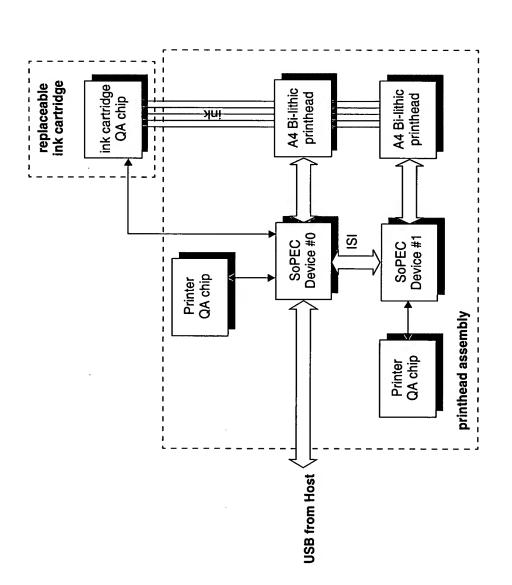
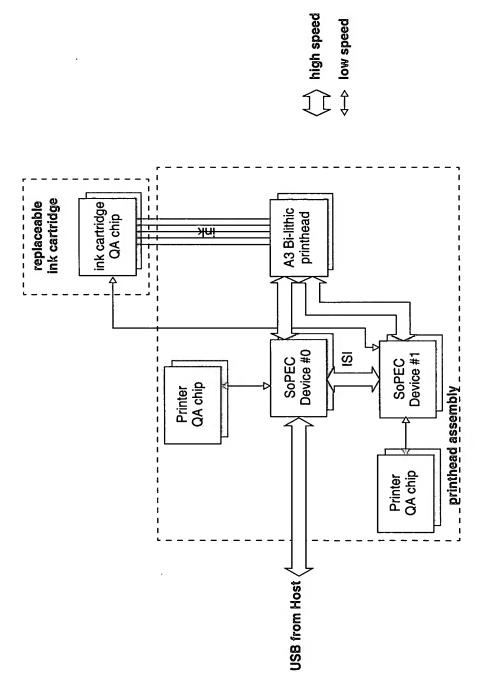


FIG. 4



F1G. 5

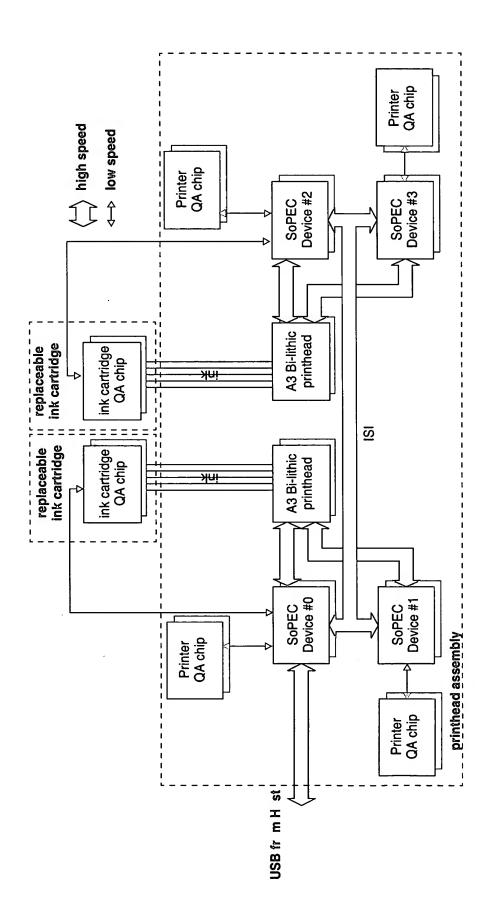


FIG. 6

high speed ←

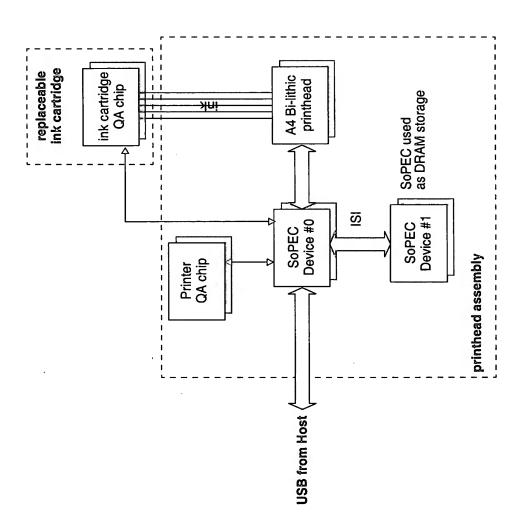


FIG. 7

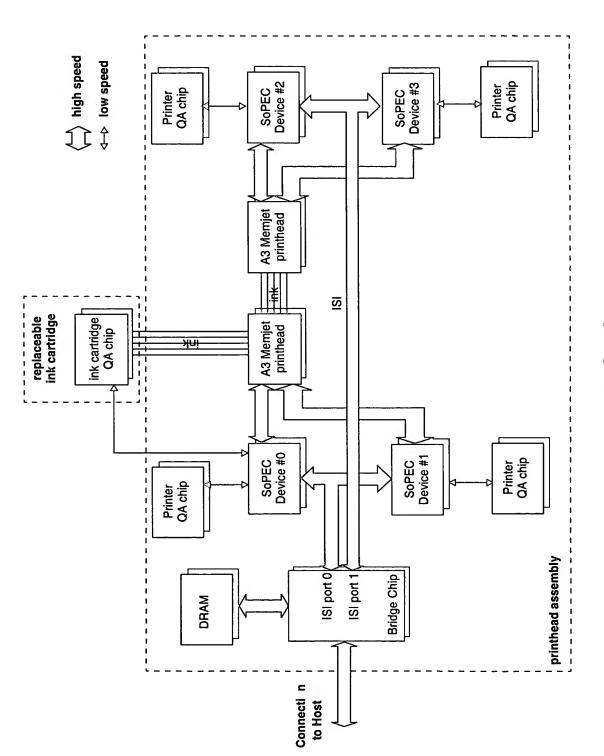
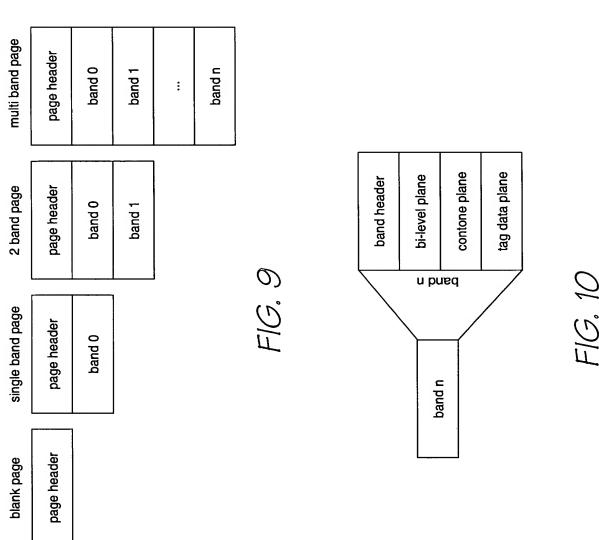


FIG. 8



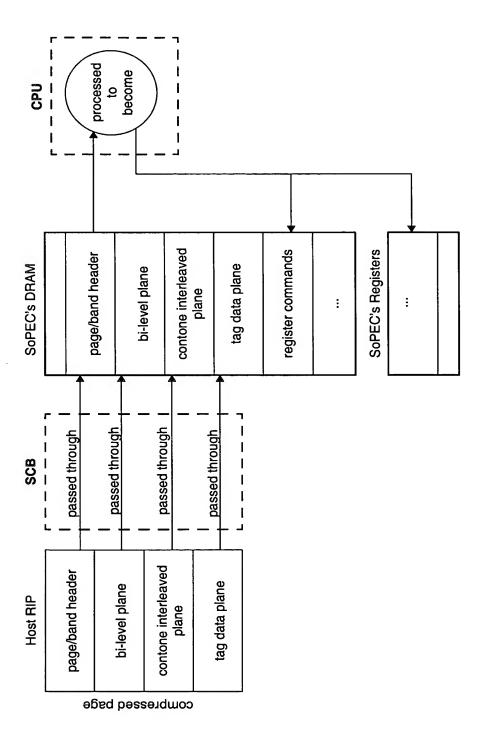


FIG. 11

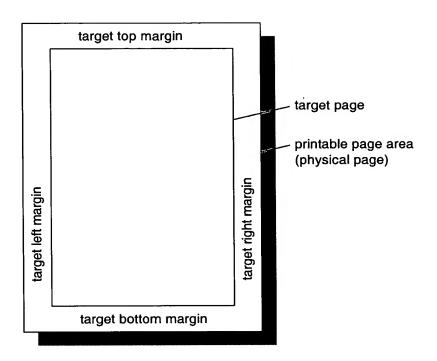


FIG. 12

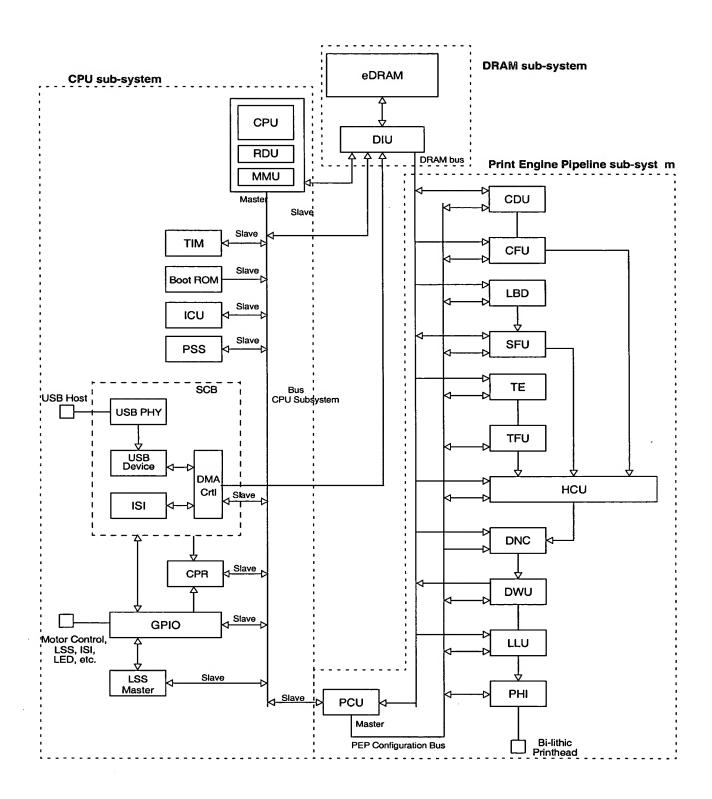
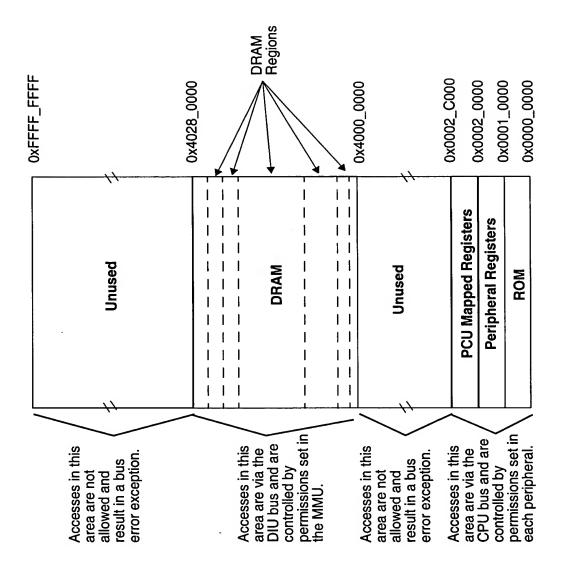


FIG. 13



F1G. 14

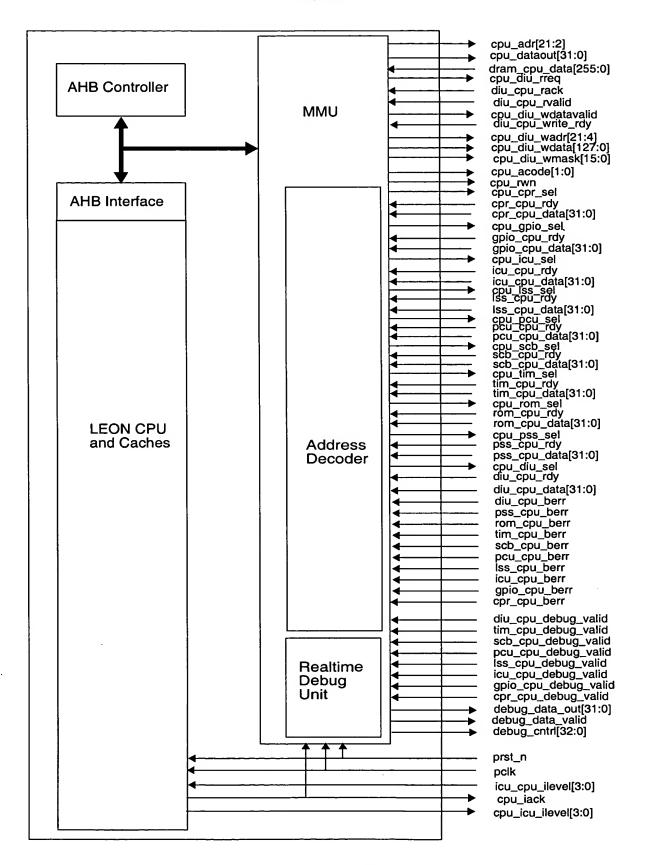


FIG. 15

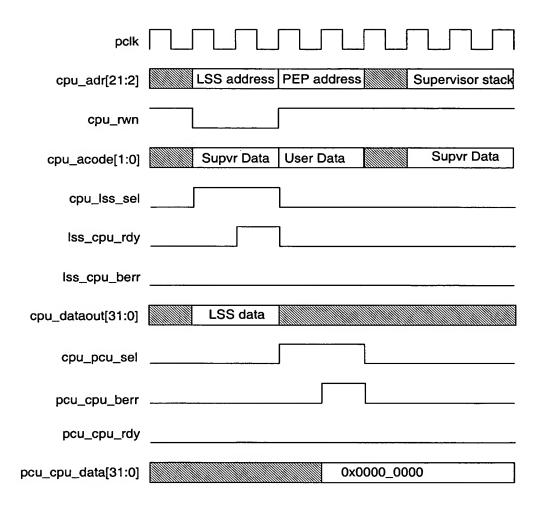


FIG. 16

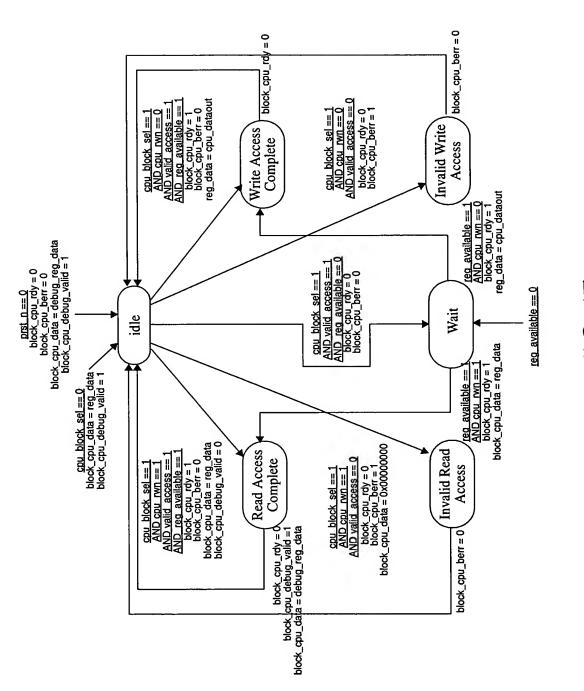


FIG. 17

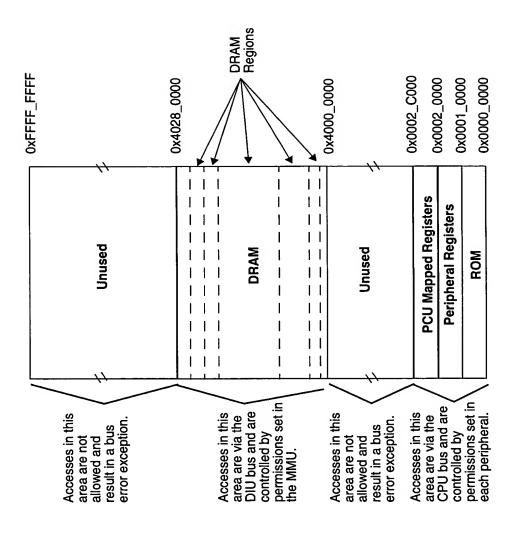


FIG. 18

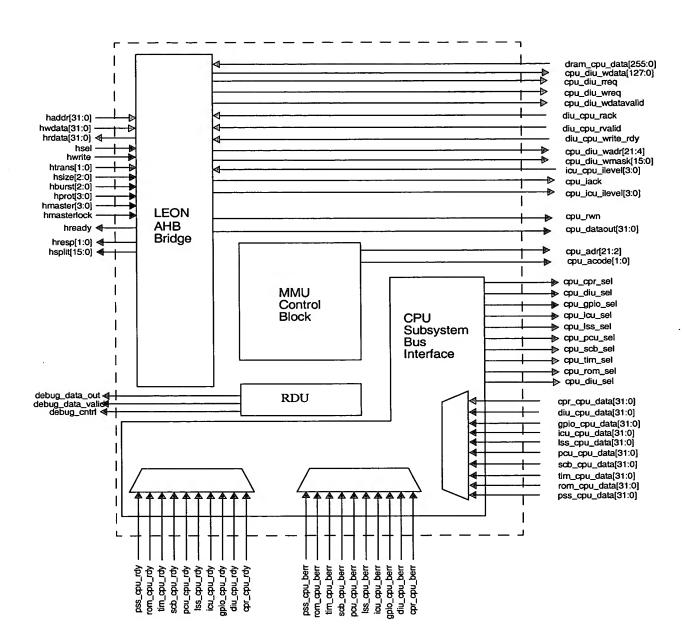
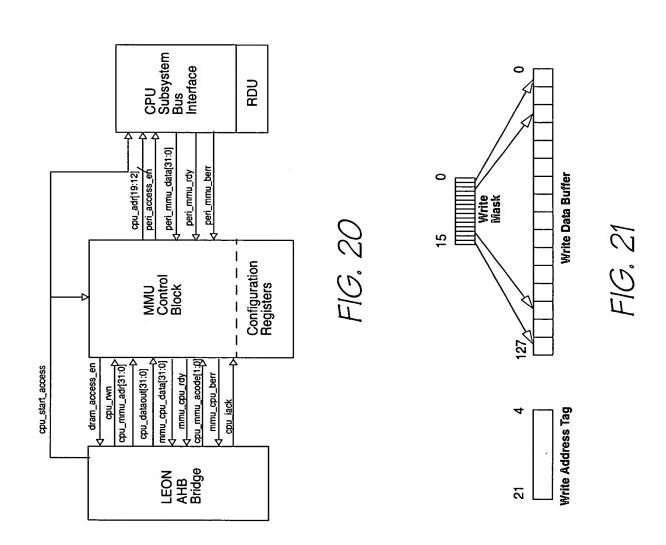


FIG. 19



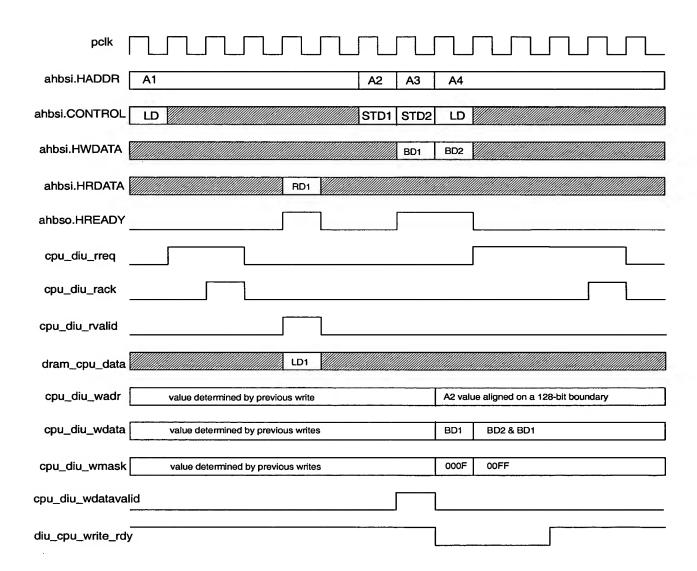


FIG. 22

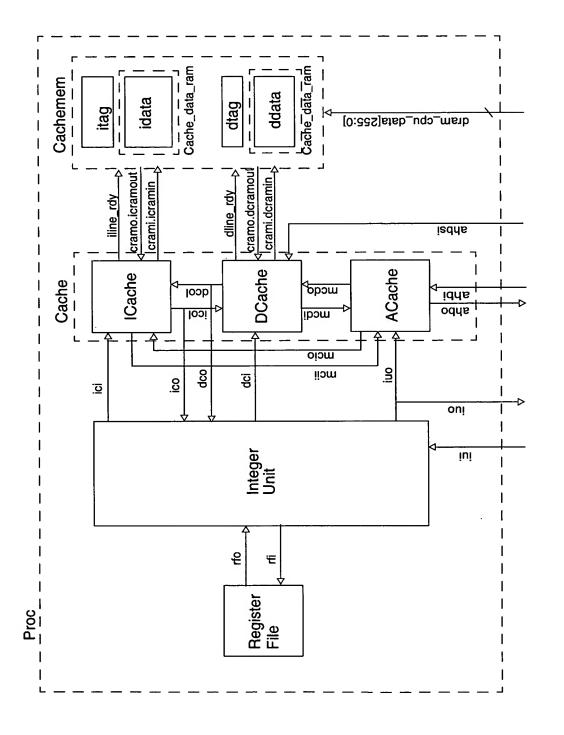
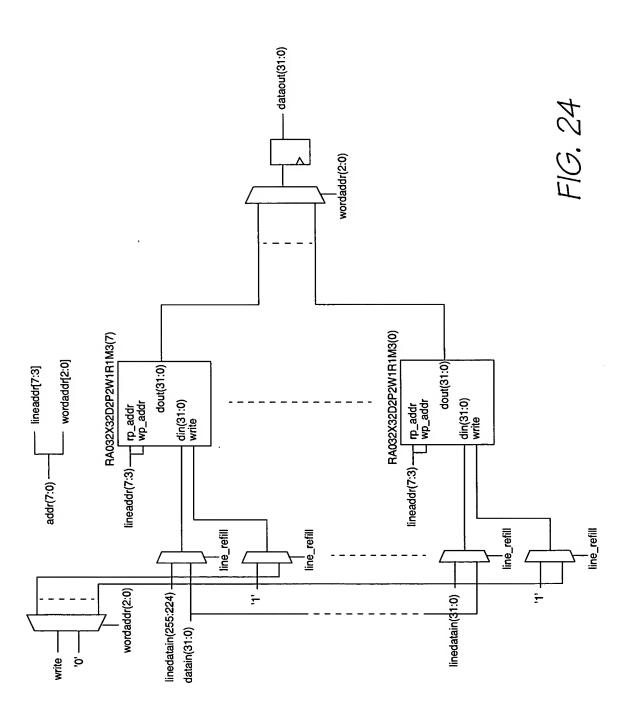
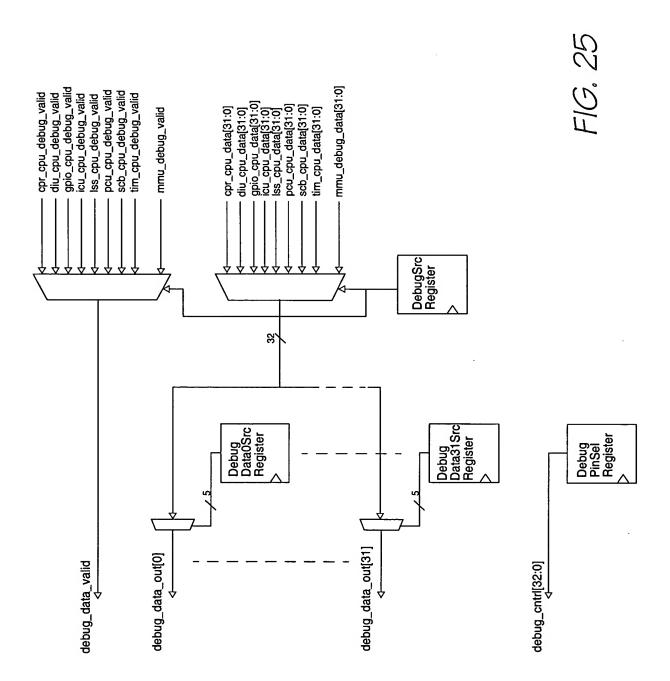


FIG. 23





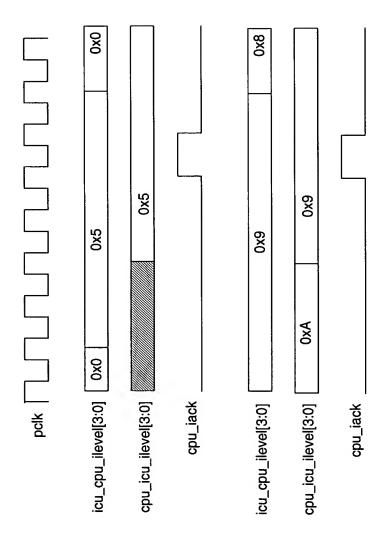


FIG. 26

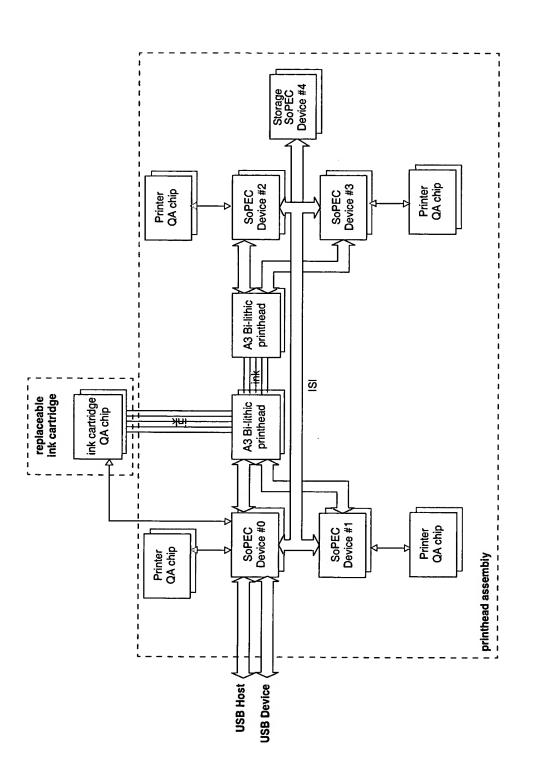


FIG. 27

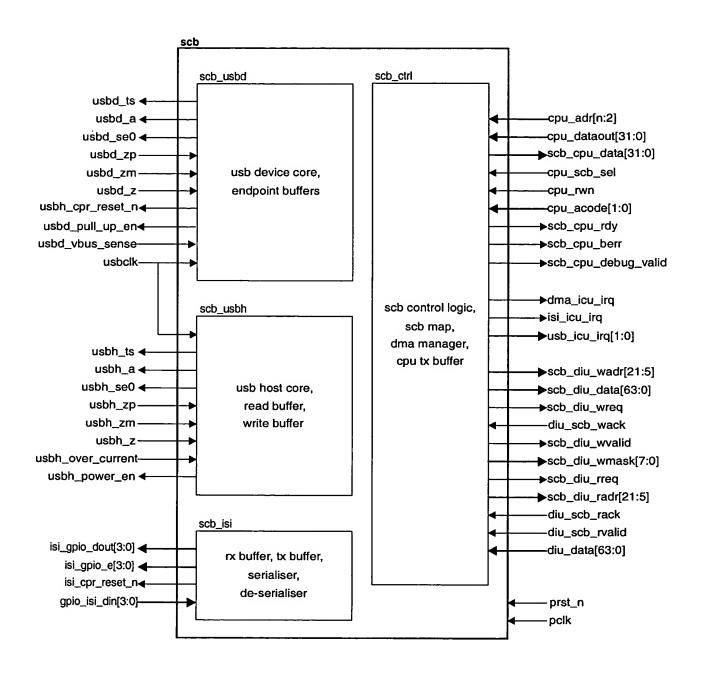


FIG. 28

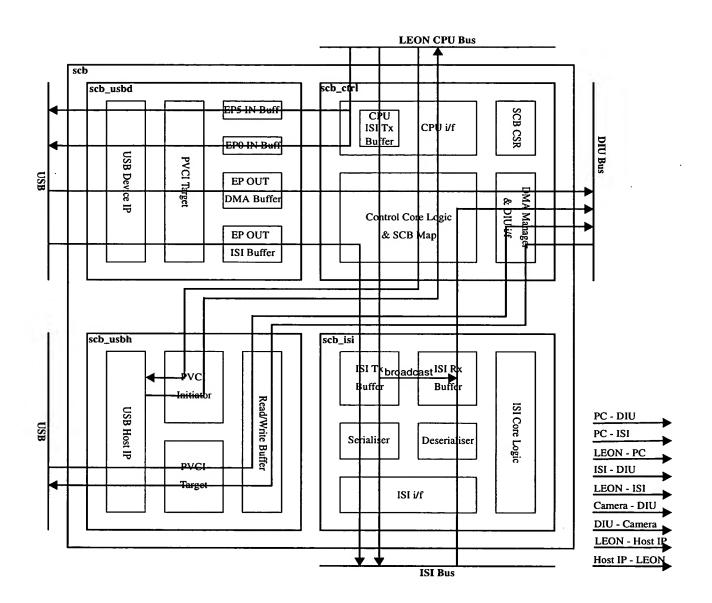


FIG. 29

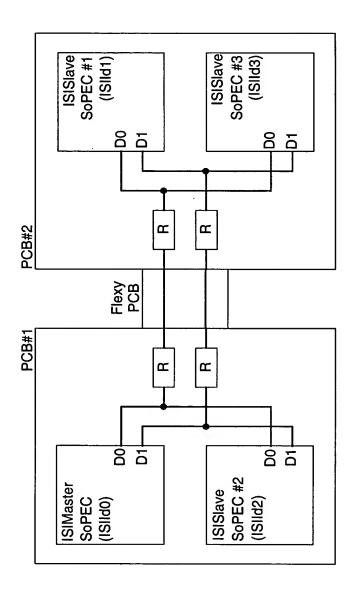
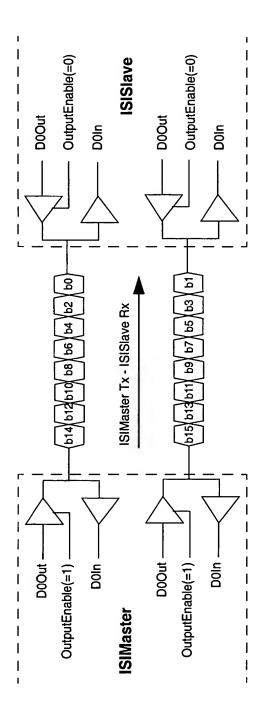
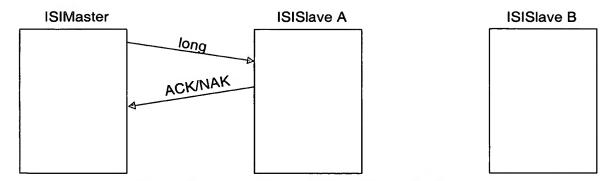


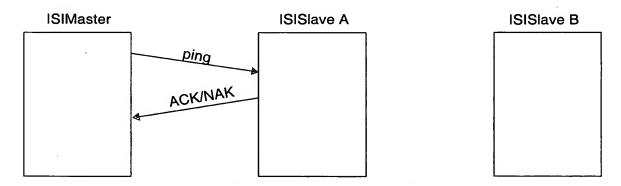
FIG. 30



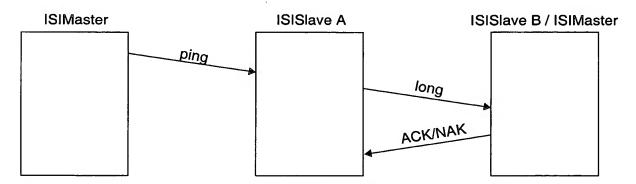
F1G. 3



Transaction 1: Long packet to an addressed ISISlave



Transaction 2: Ping packet to an addressed ISISlave. ISISlave has nothing to send



Transaction 3: Ping packet to an addressed ISISlave. ISISlaveA responds with a long packet to ISISlaveB (or the ISIMaster) and ISISlaveB (or the ISIMaster) responds with an ACK or NAK.

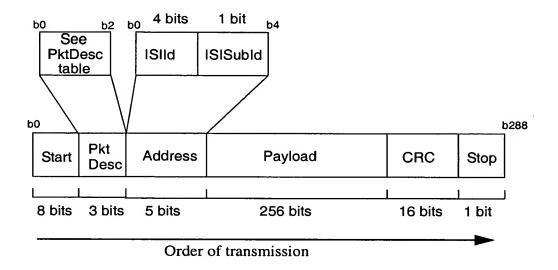


FIG. 33

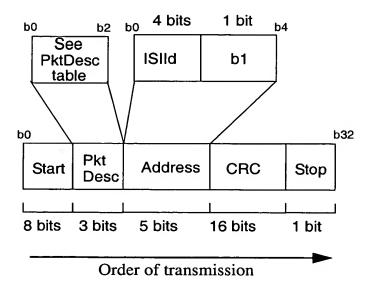


FIG. 34

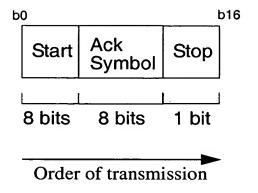


FIG. 35

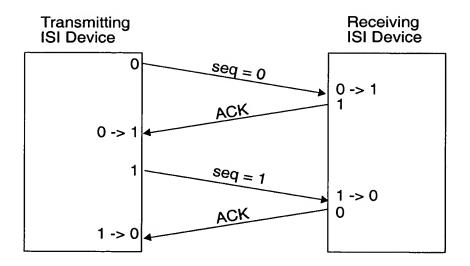


FIG. 36

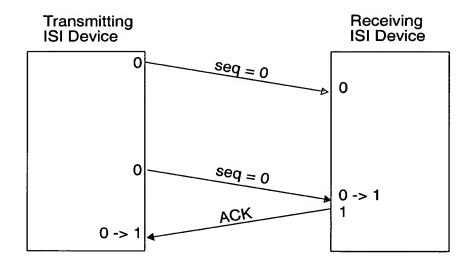


FIG. 37

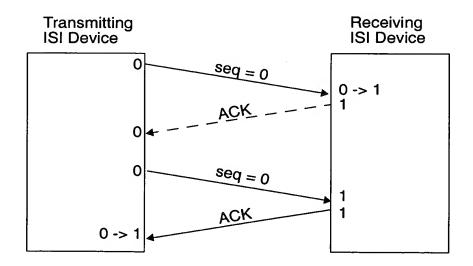


FIG. 38

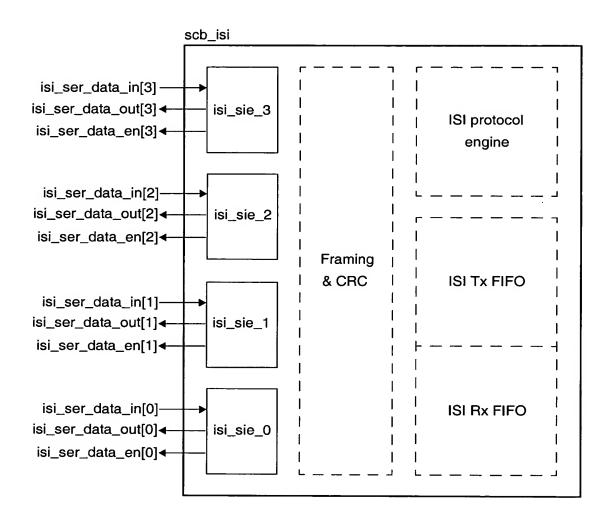


FIG. 39

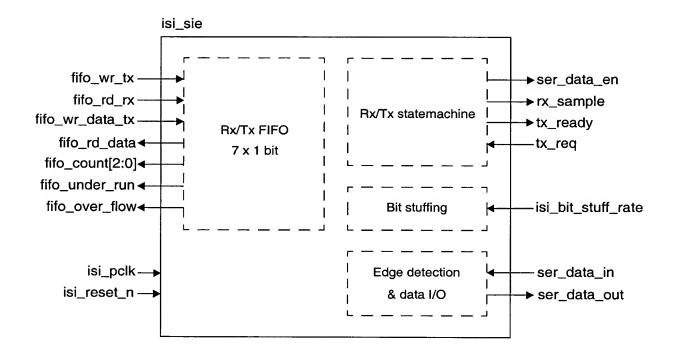


FIG. 40

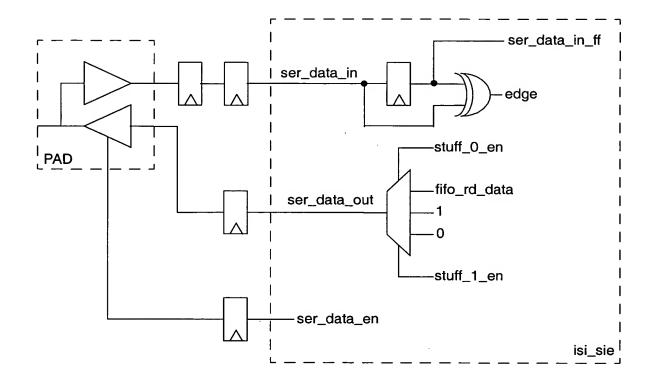


FIG. 41

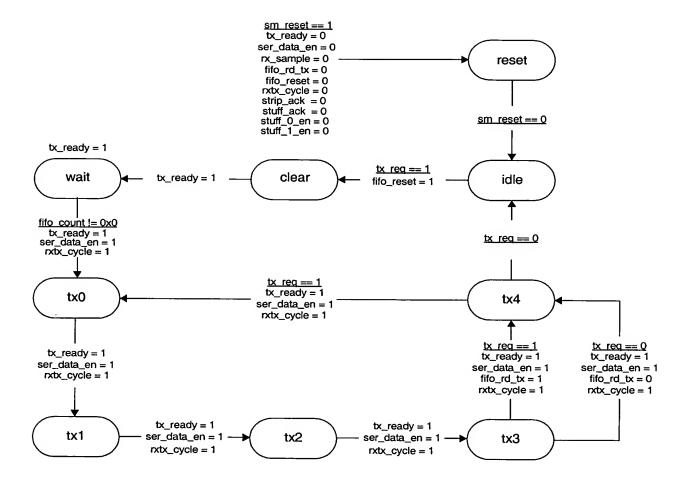


FIG. 42

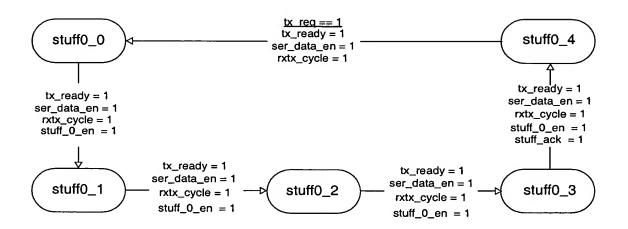


FIG. 43

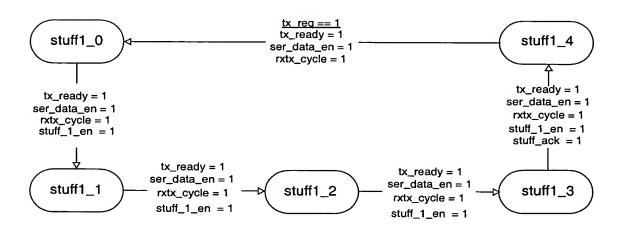


FIG. 44

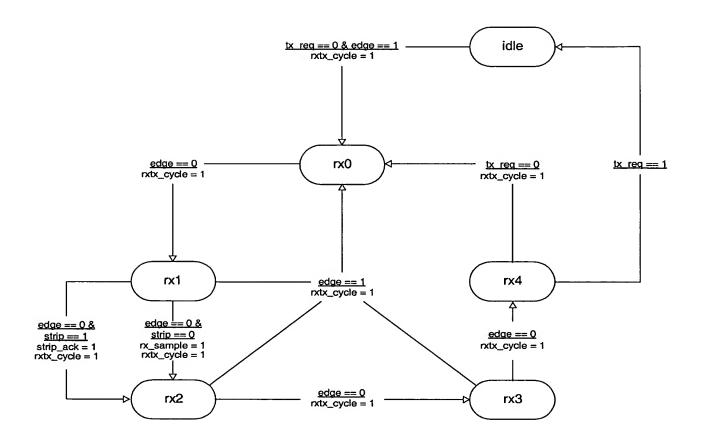


FIG. 45

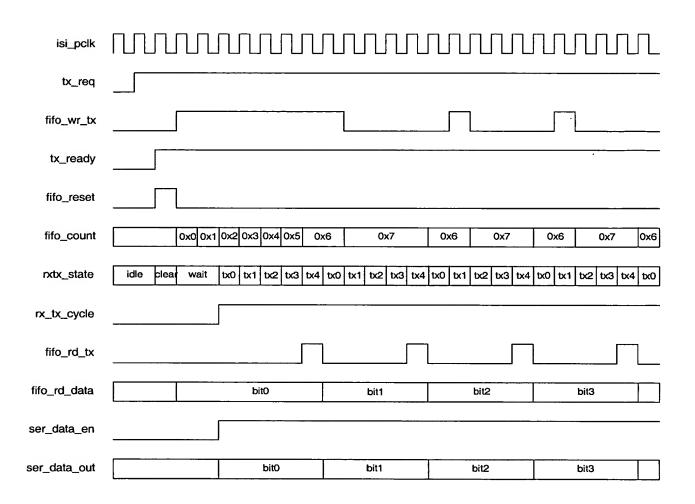


FIG. 46

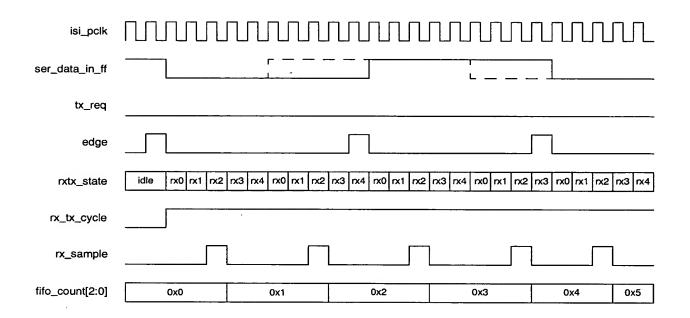


FIG. 47

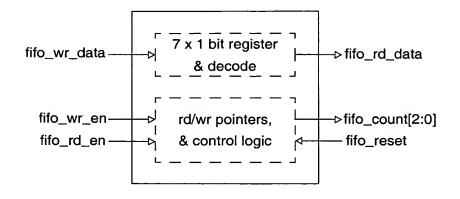


FIG. 48

FIG. 49

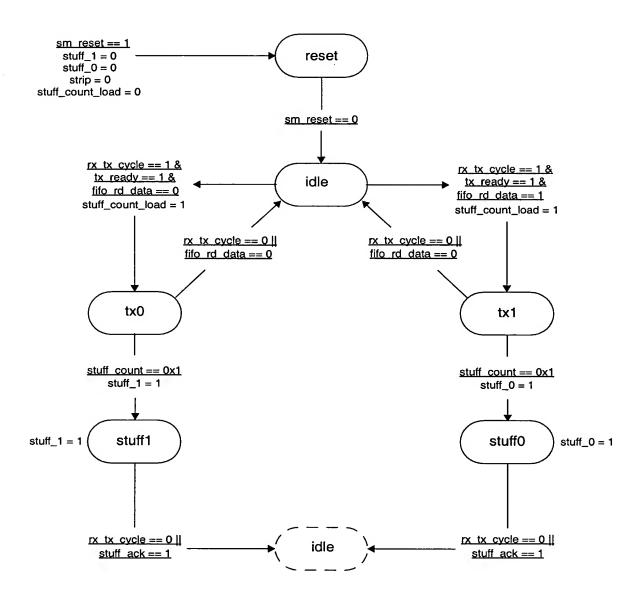


FIG. 50

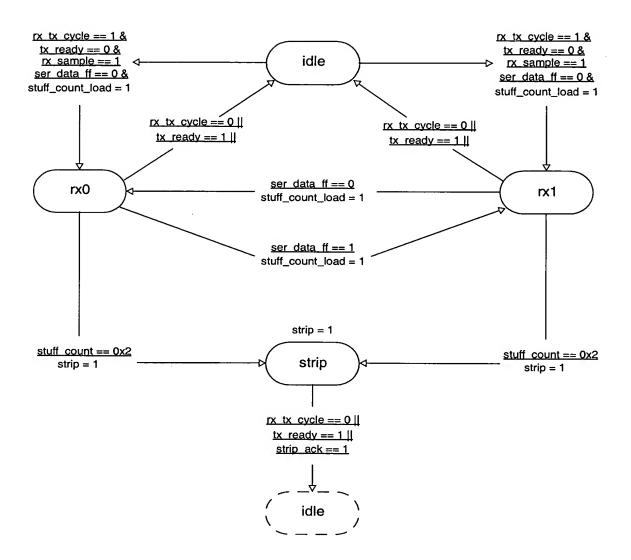


FIG. 51

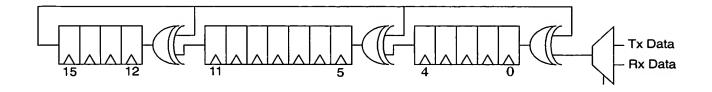


FIG. 52

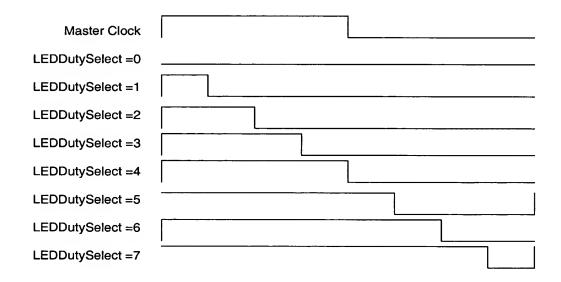


FIG. 54

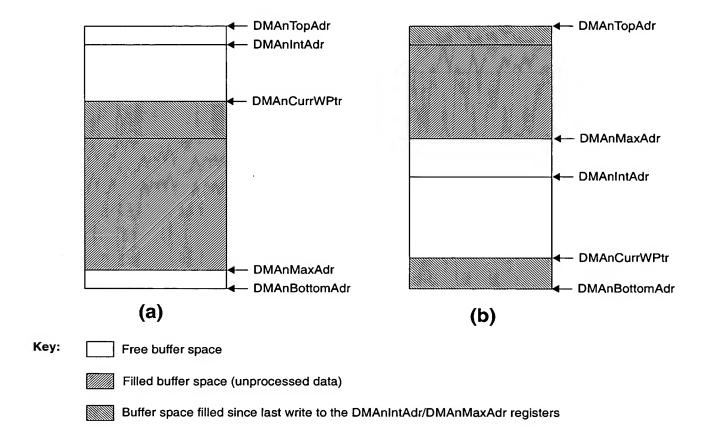


FIG. 53

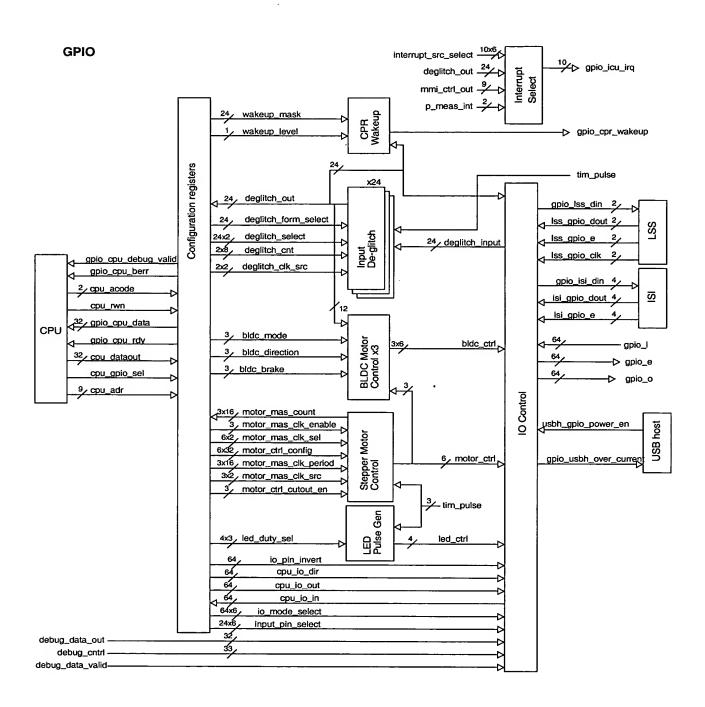
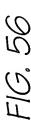
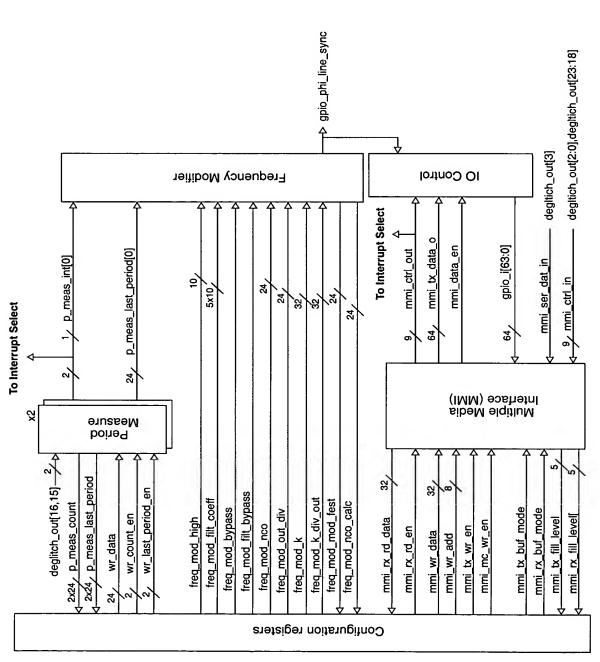


FIG. 55





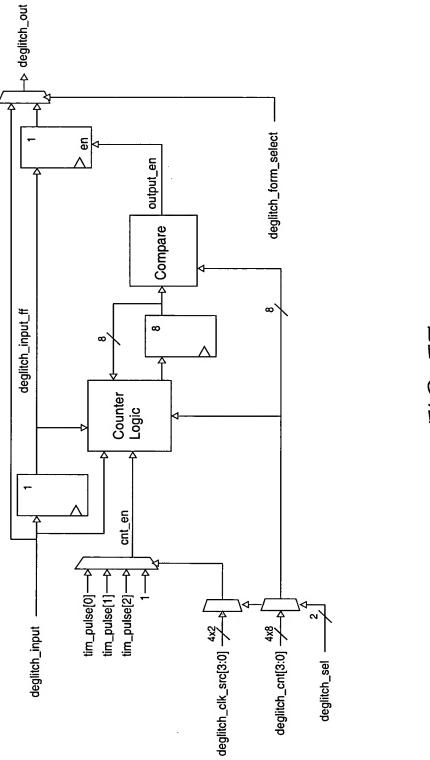


FIG. 57

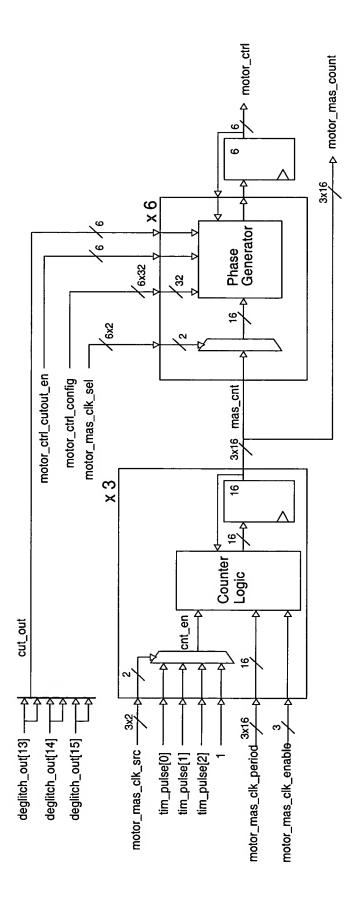
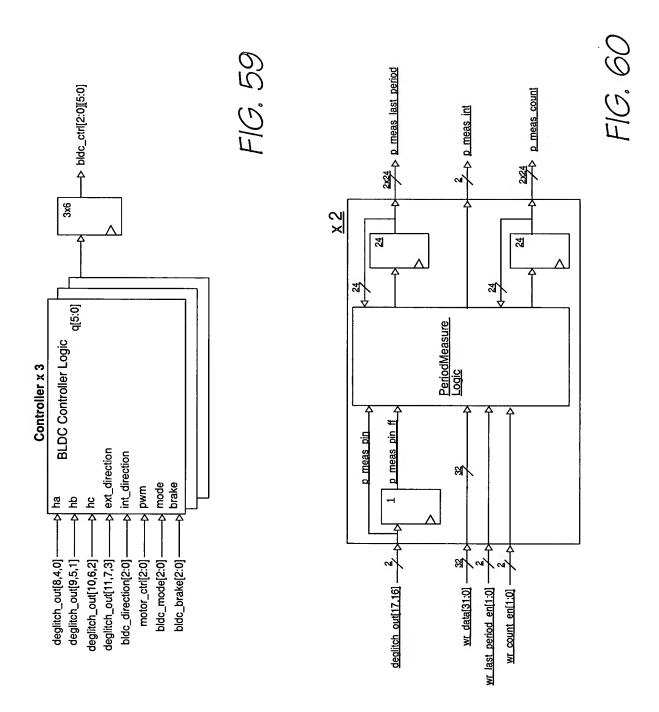


FIG. 58



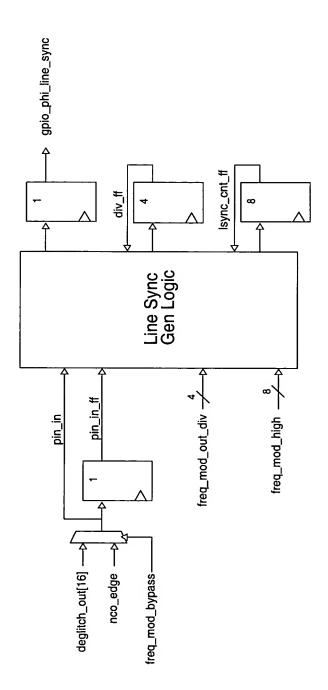


FIG. 61

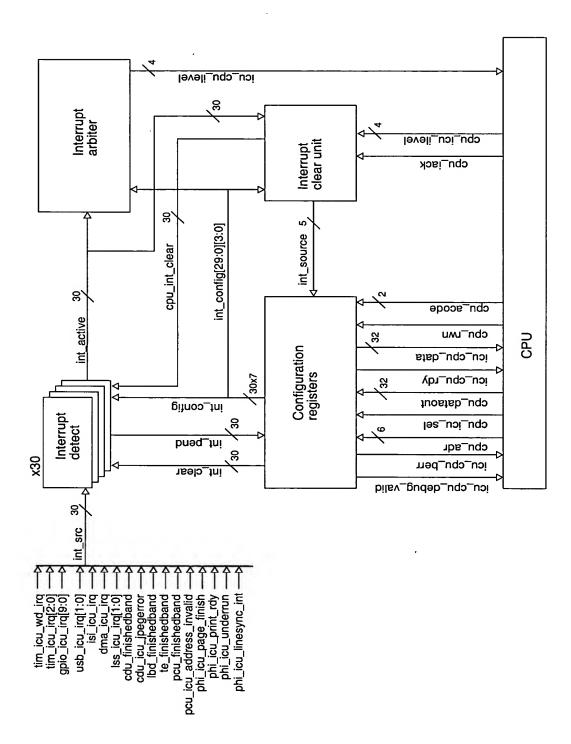


FIG. 62



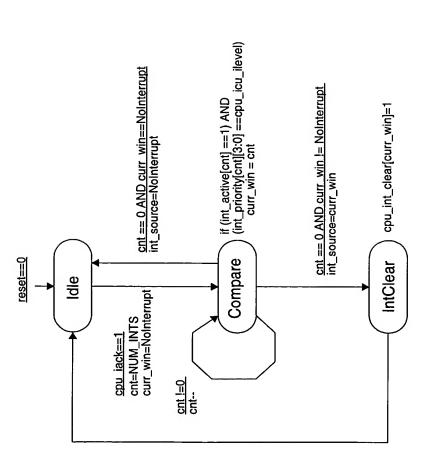
Machine remains in same state by default All outputs are zero unless otherwise stated

State Description:

Normal Idle state Idje: Compare interrupt level. Determines the Compare:

interrupt source

Interrupt clear, clear the pending bit for the current interrupt vector IntClear:



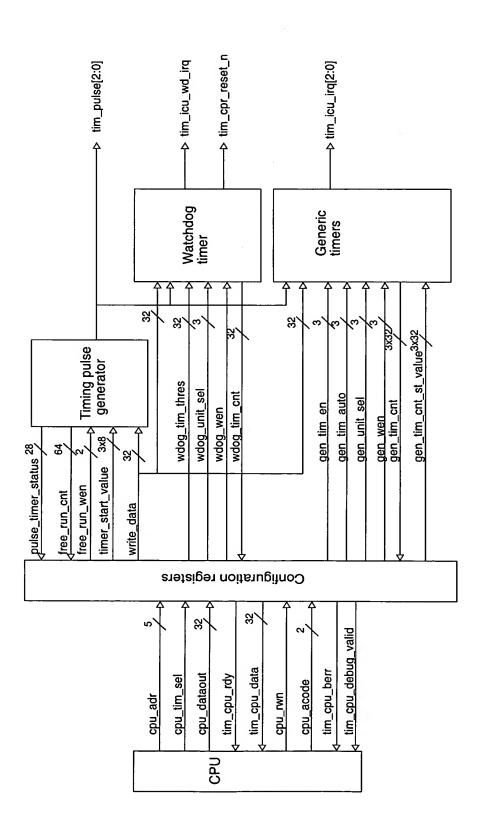
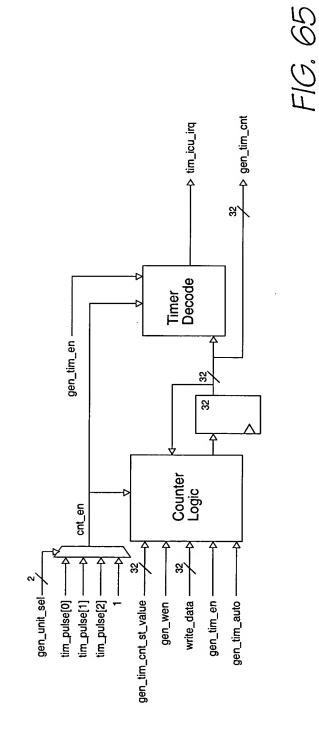


FIG. 63A



Timer Decode

8,

32

Counter Logic

8

wdog_wen write_data —

wdog_tim_thres -----

cnt_en

wdog_unit_sel___

tim_pulse[0] -

tim_pulse[1] -tim_pulse[2] --

FIG. 64

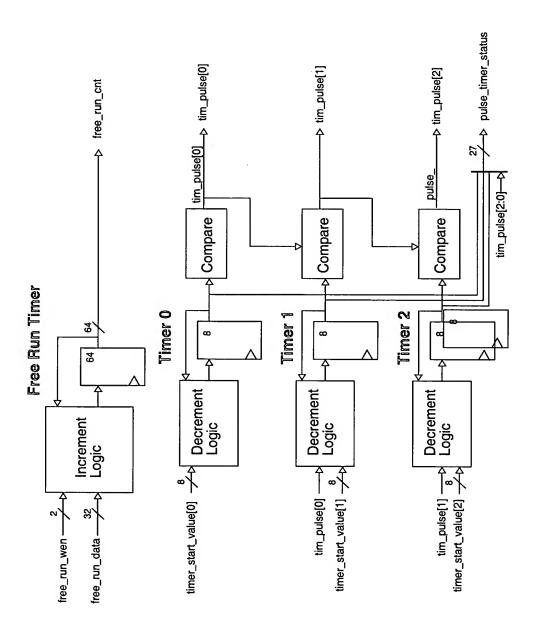


FIG. 66

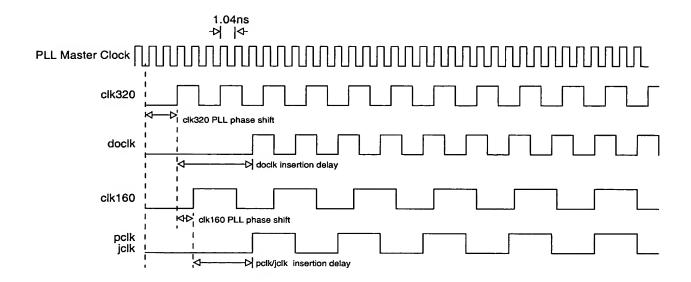


FIG. 67

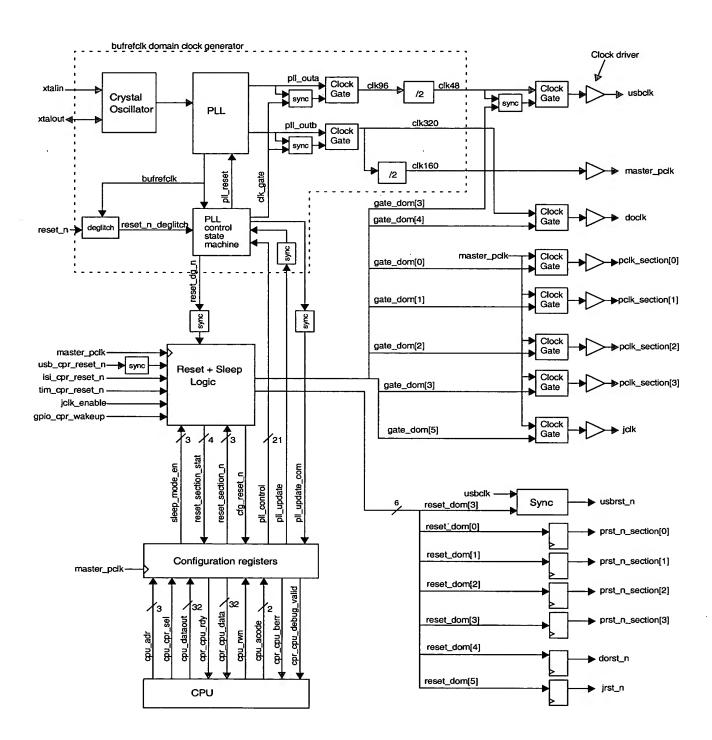
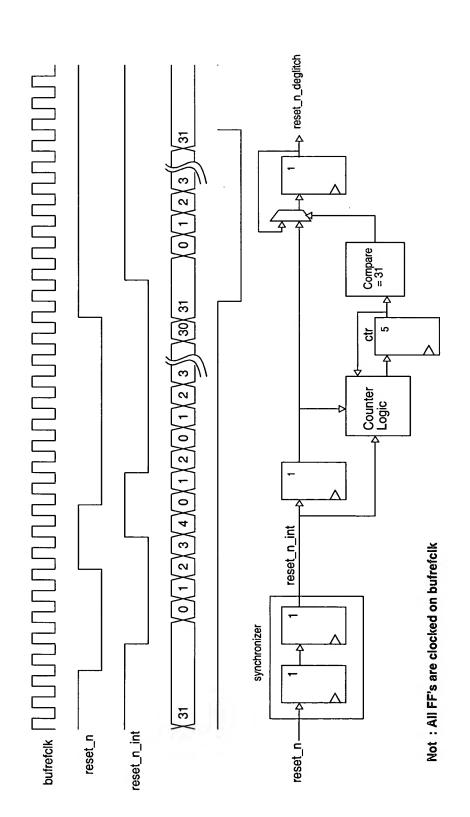
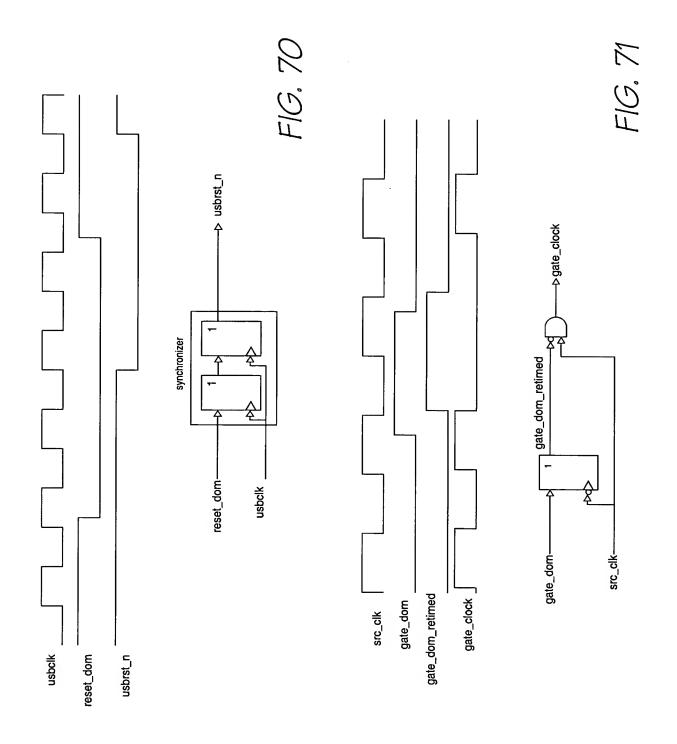
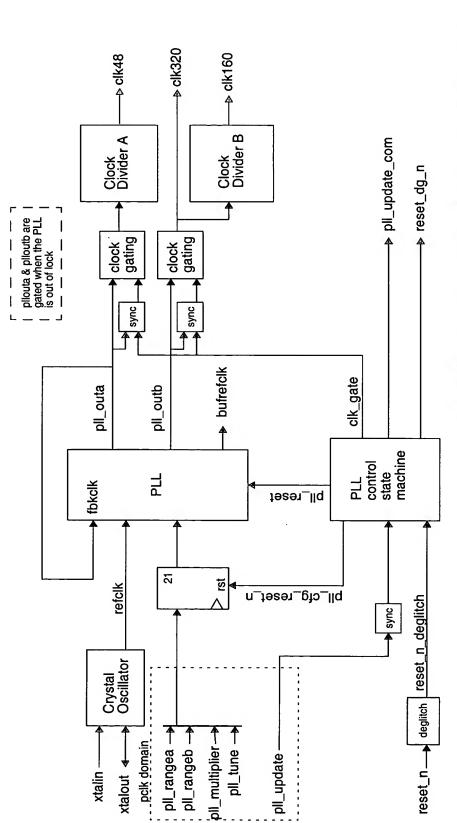


FIG. 68







Note: All logic clocked on bufrefclk unless otherwise indicated

FIG. 72

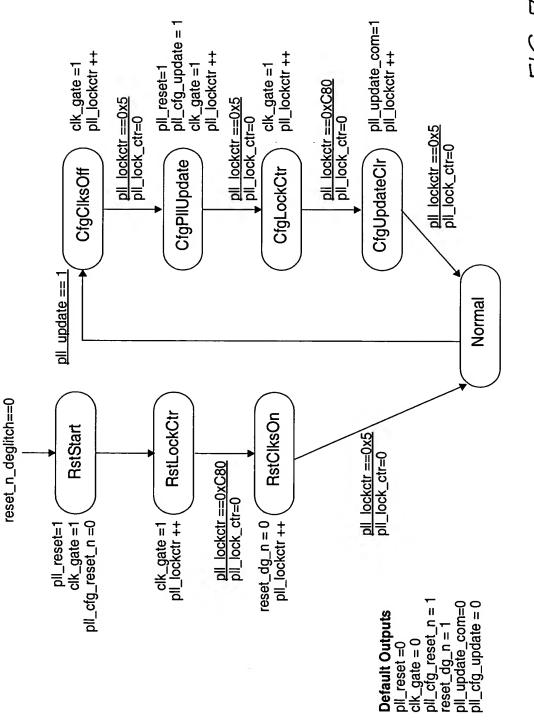
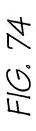
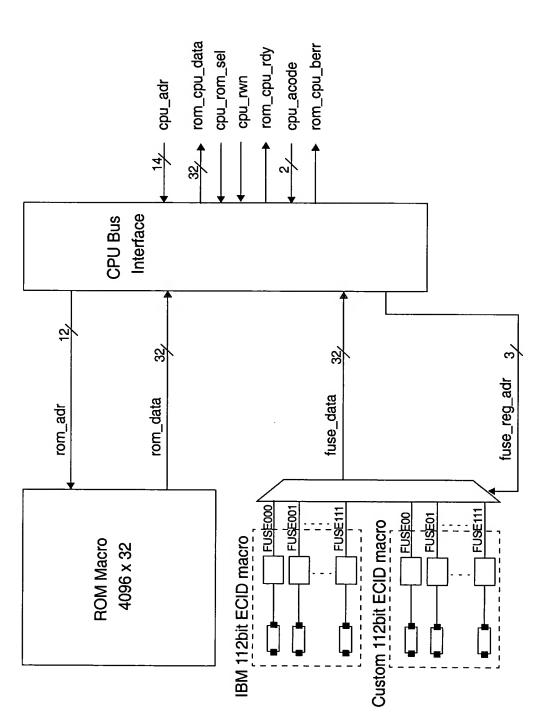


FIG. 73





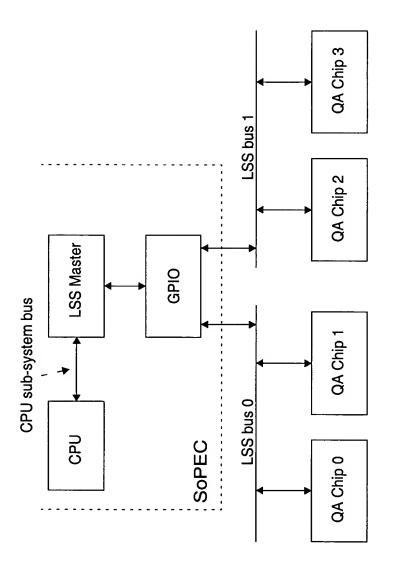
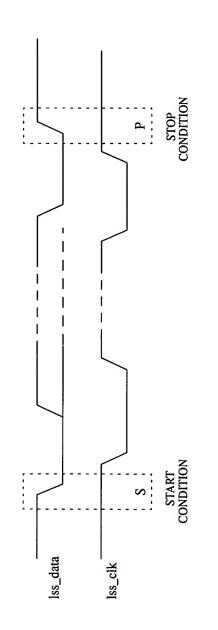


FIG. 75



F1G, 76

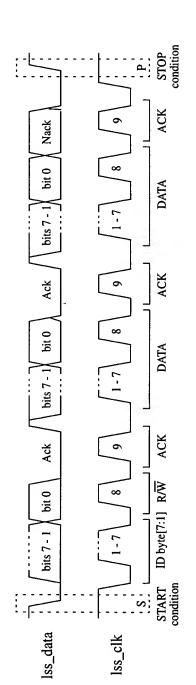


FIG. 77

Ь A Data(8) A Byte M-1 🗘 Data(8) A Byte 0 A Data(8) 0 D byte[7:1] S

FIG. 78

S = Start condition
A = Ack
N = Nack
P = Stop condition
Shaded bits driven by slave

S = Start condition
A = Ack
N = Nack
P = Stop condition
Shaded bits driven by slave

Д

Byte M

Byte 1

Byte 0

S

Z ---- Data(8) 4 Data(8) A Data(8) A ID byte[7:1]

FIG. 79

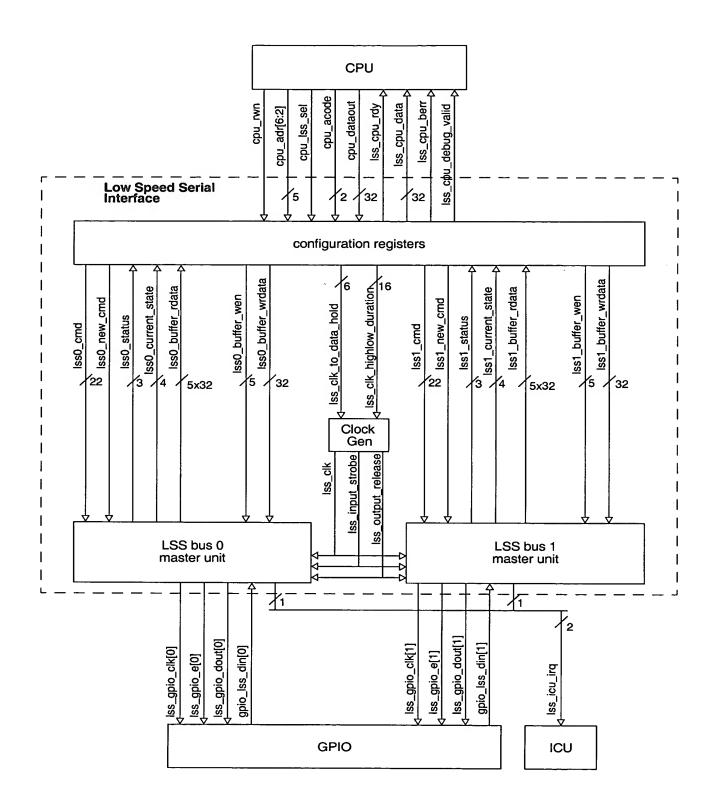


FIG. 80

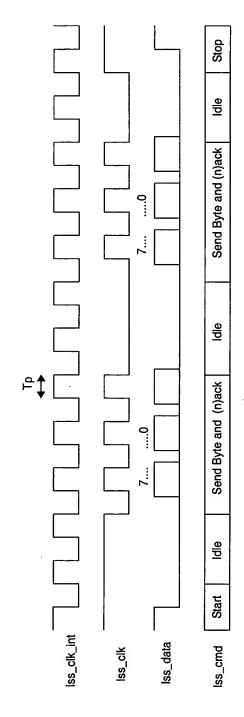


FIG. 81

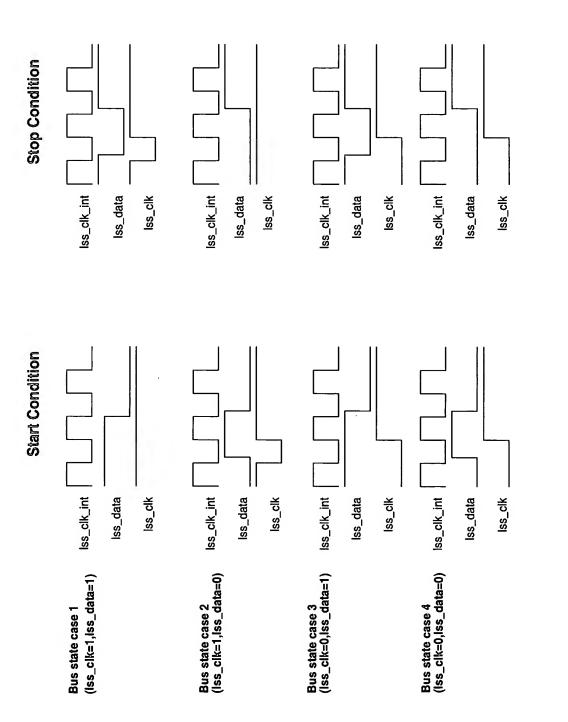


FIG. 82

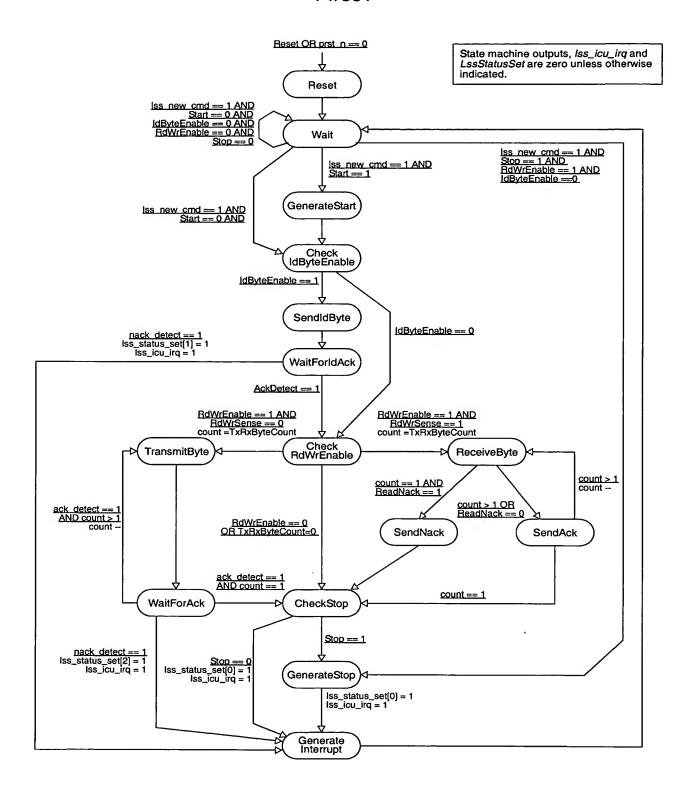


FIG. 83

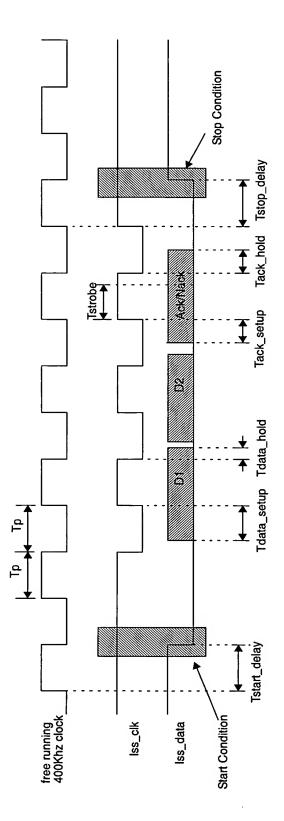


FIG. 84

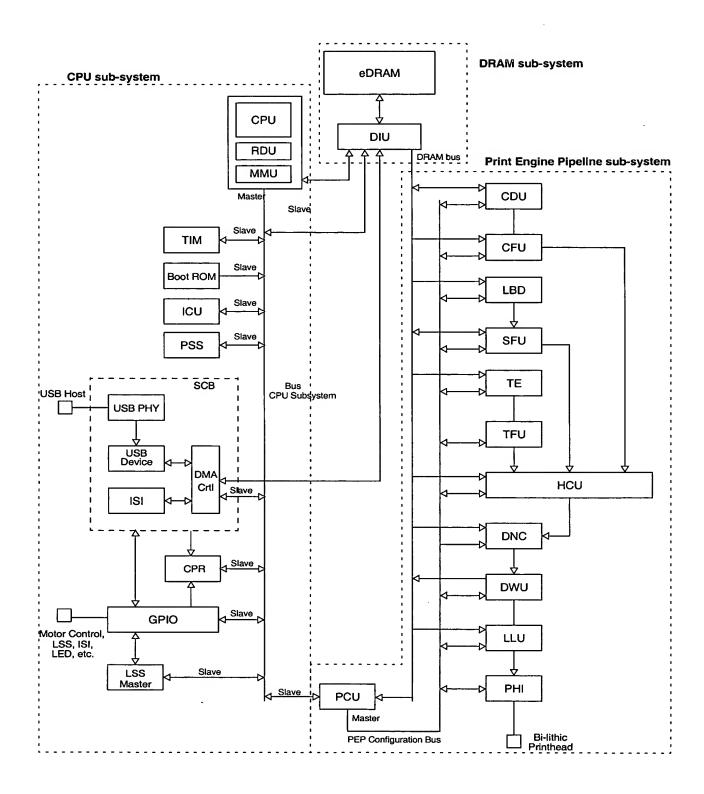
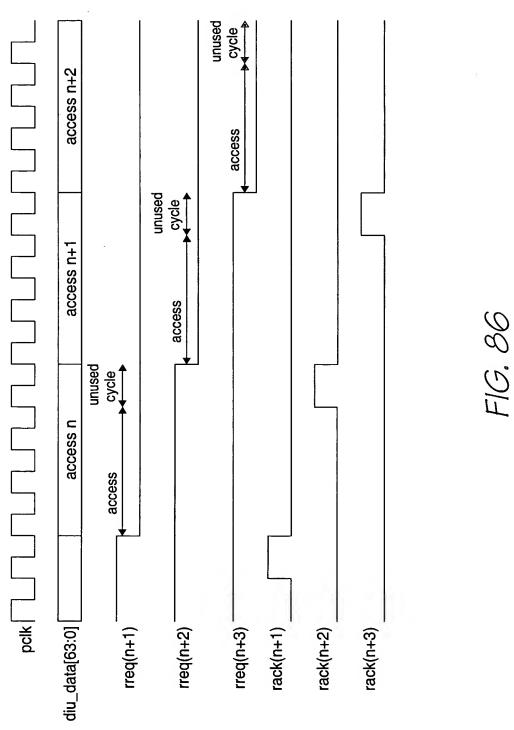


FIG. 85



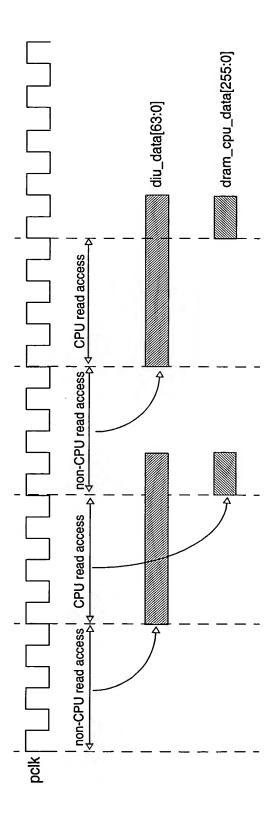


FIG. 87

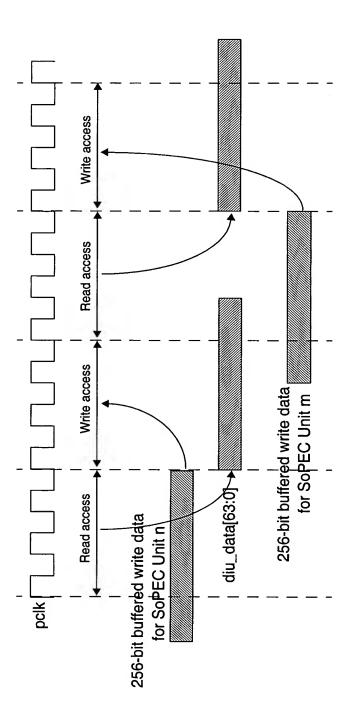


FIG. 88

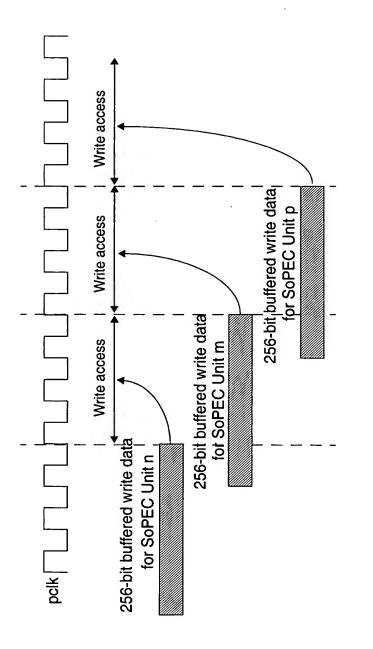


FIG. 89

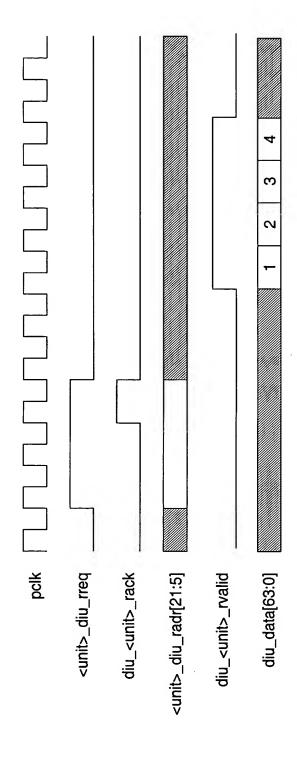


FIG. 90

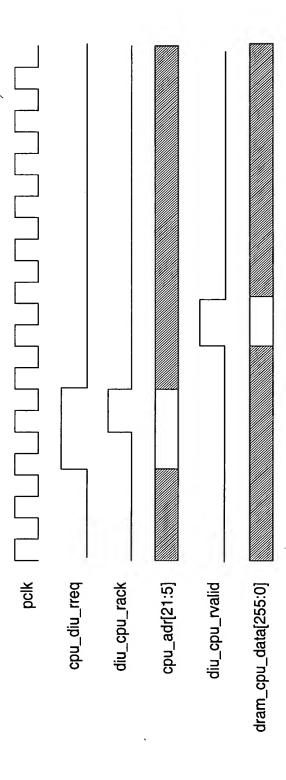


FIG. 91

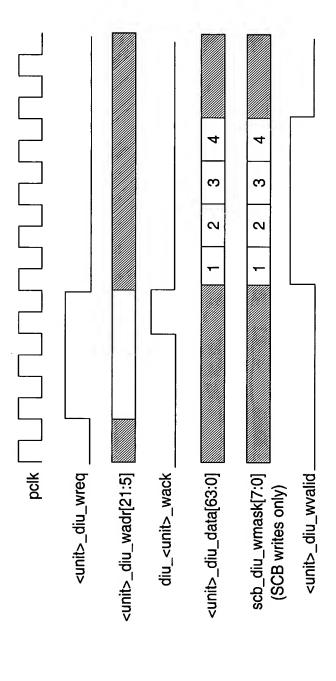


FIG. 92

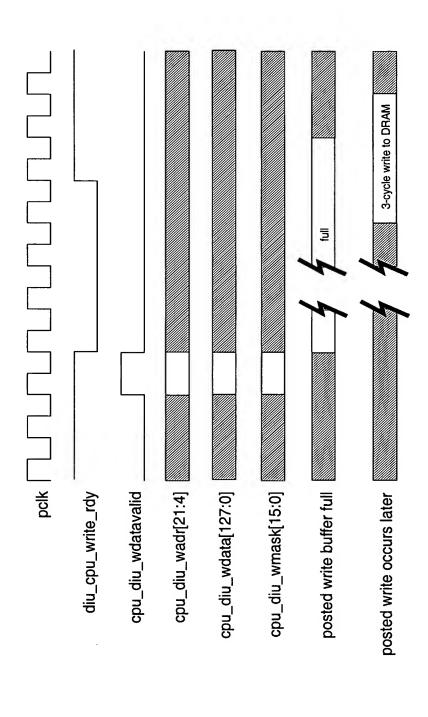


FIG. 93

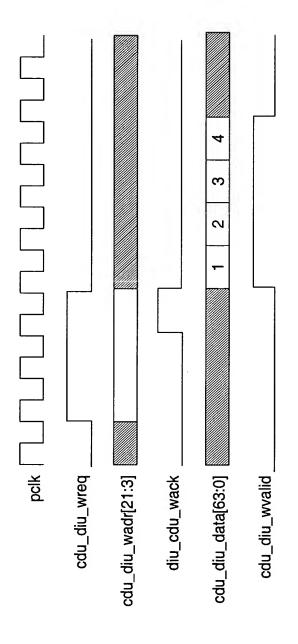


FIG. 94

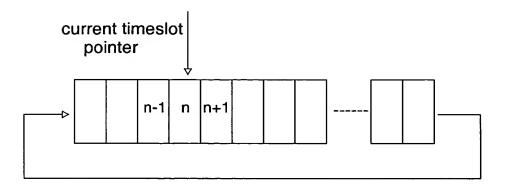


FIG. 95

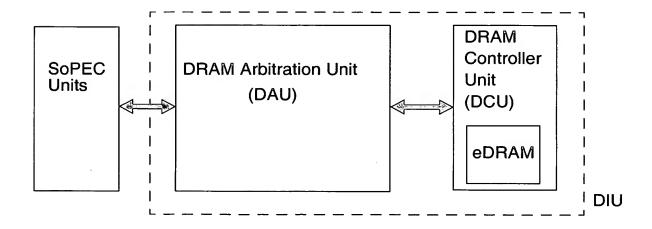


FIG. 100

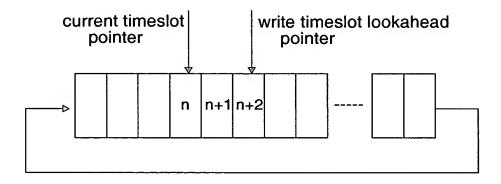


FIG. 96

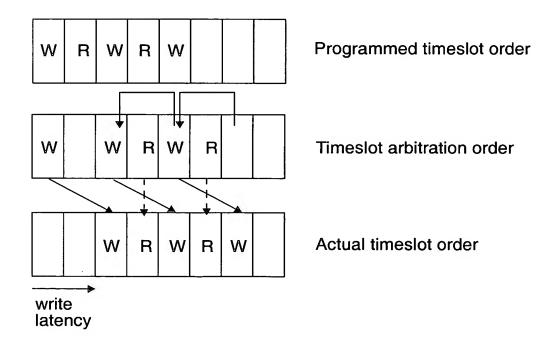


FIG. 97

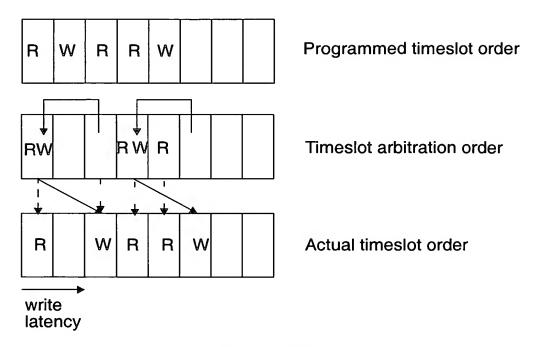


FIG. 98

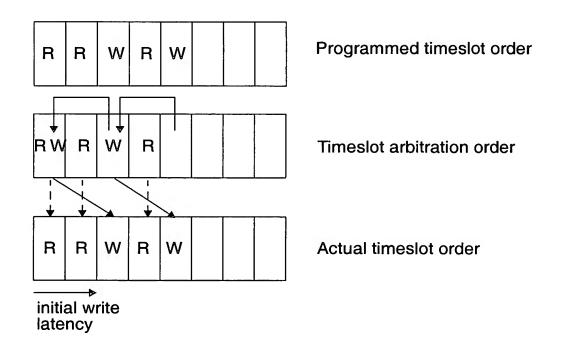


FIG. 99

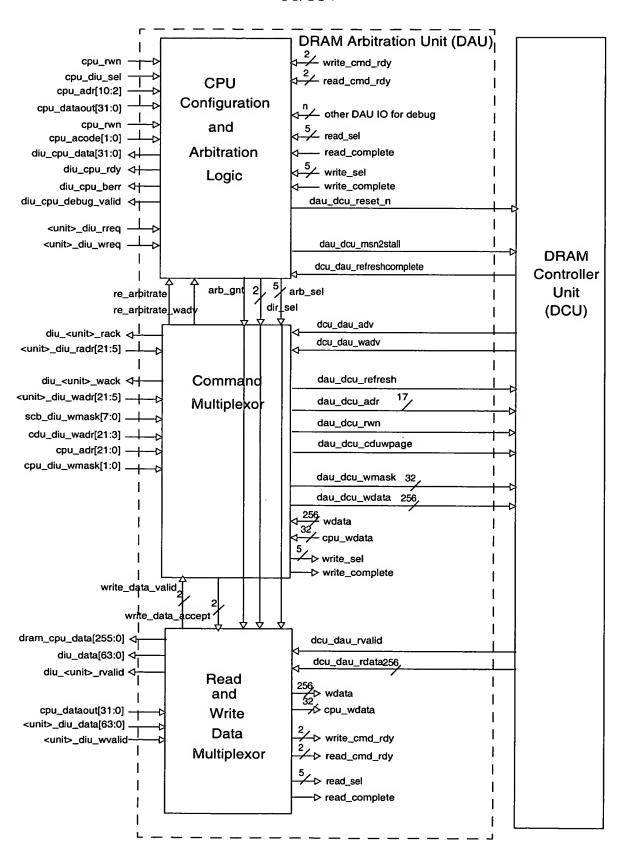


FIG. 101

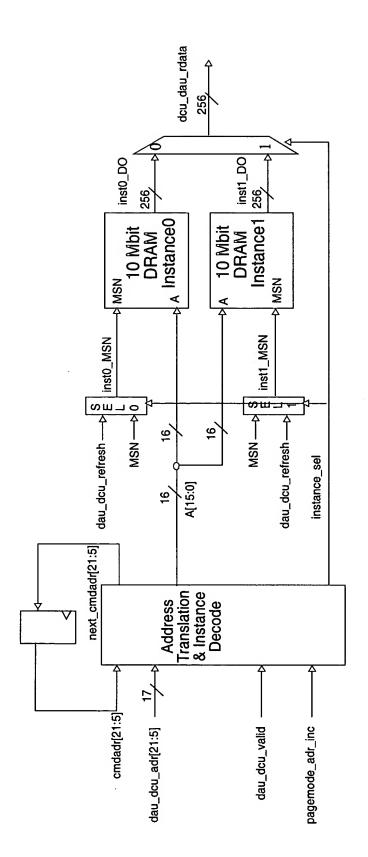


FIG. 102

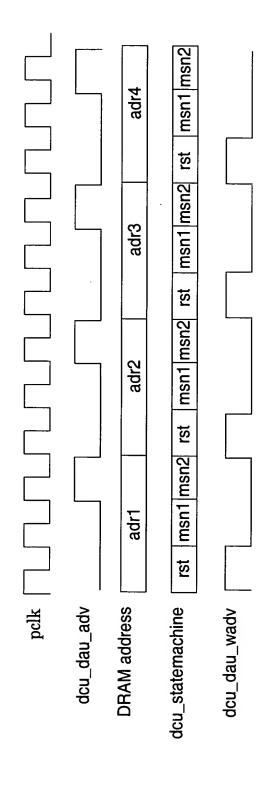


FIG. 103

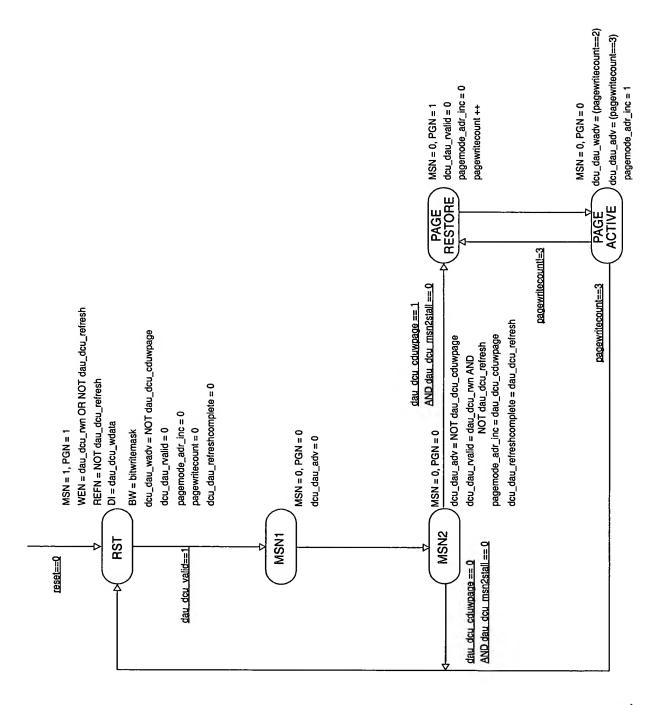


FIG. 104

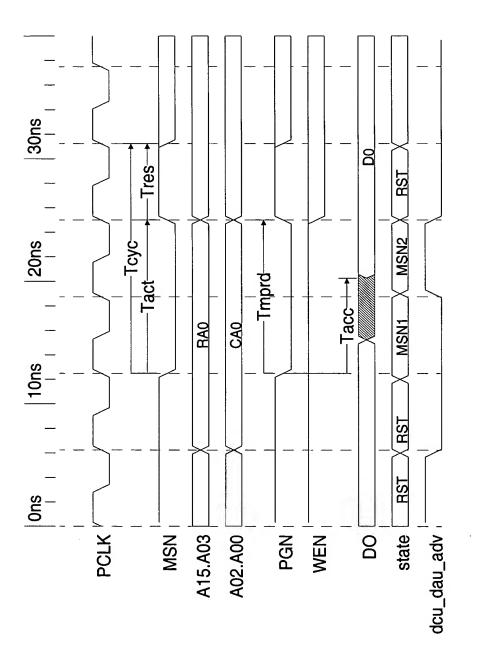


FIG. 105

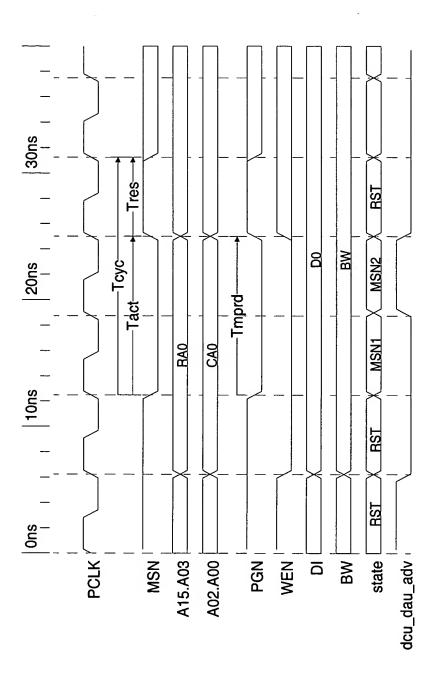


FIG. 106

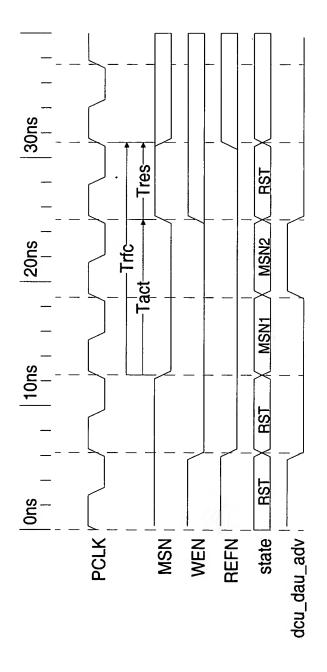


FIG. 107

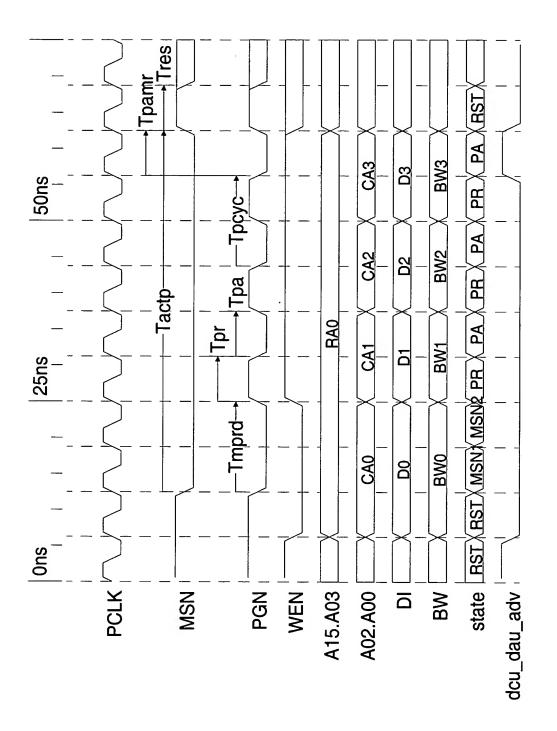


FIG. 108

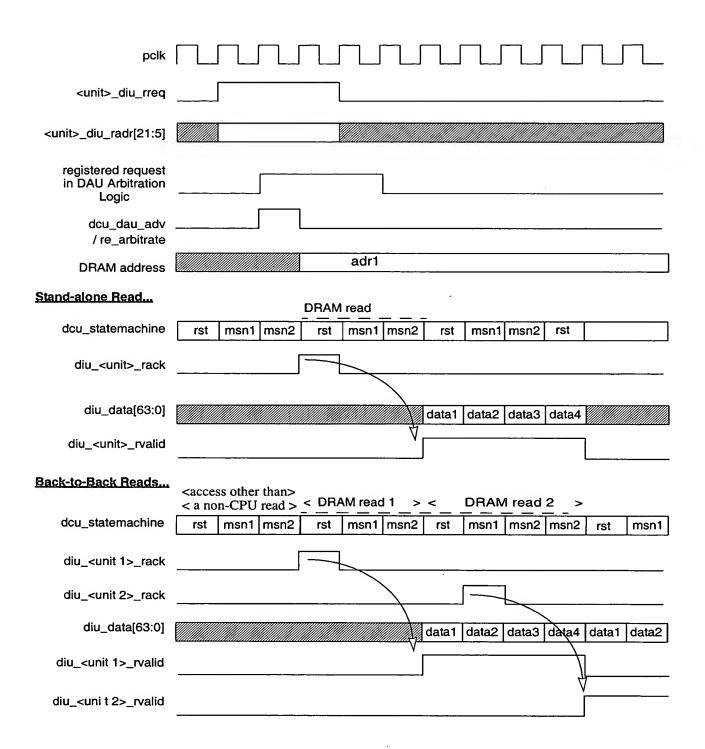


FIG. 109

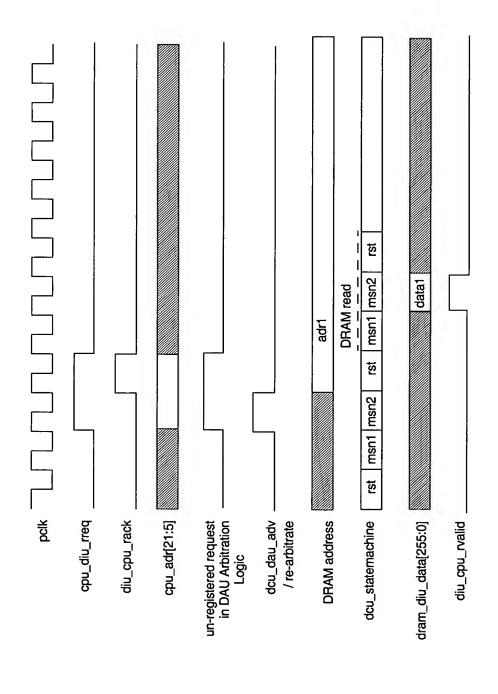
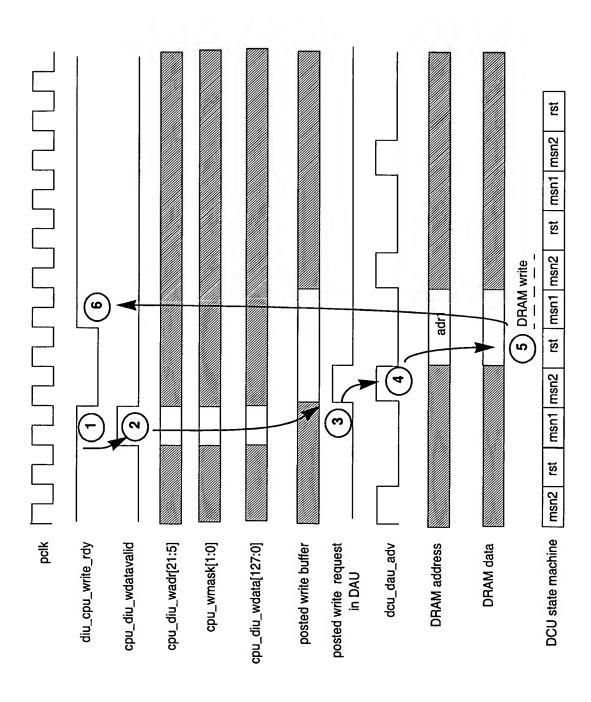
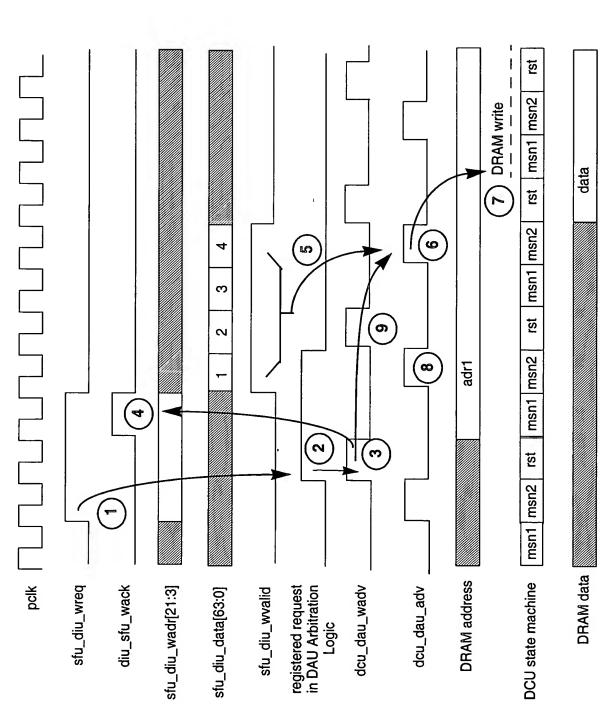


FIG. 110

CPU MMU logic delay CPU generates request Arbitration occurs	cycle 6
CPU captures read data in AHB bridge	cycle 5
DRAM access (MSN2)	cycle 4
DRAM access (MSN1) Tacc = 8 ns	cycle 3
DCU Address setup cycle (RST)	cycle 2
CPU MMU logic delay CPU generates request Arbitration occurs	cycle 1

F1G. 11





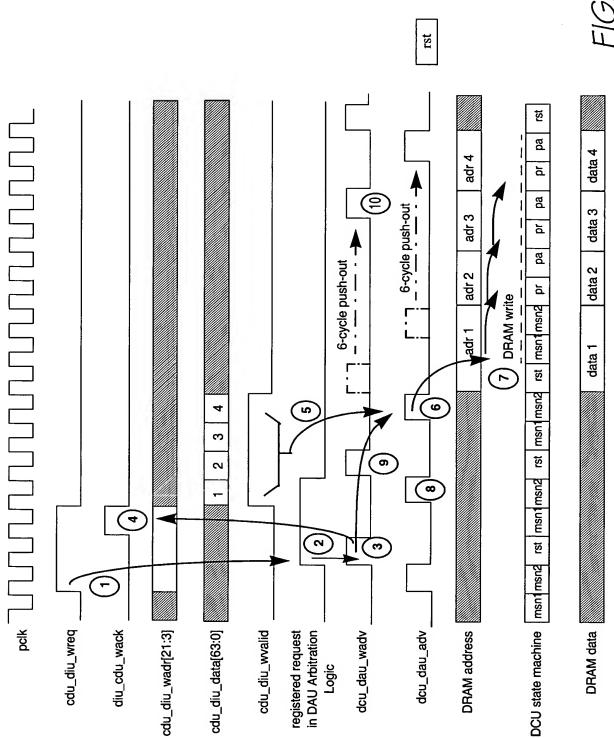
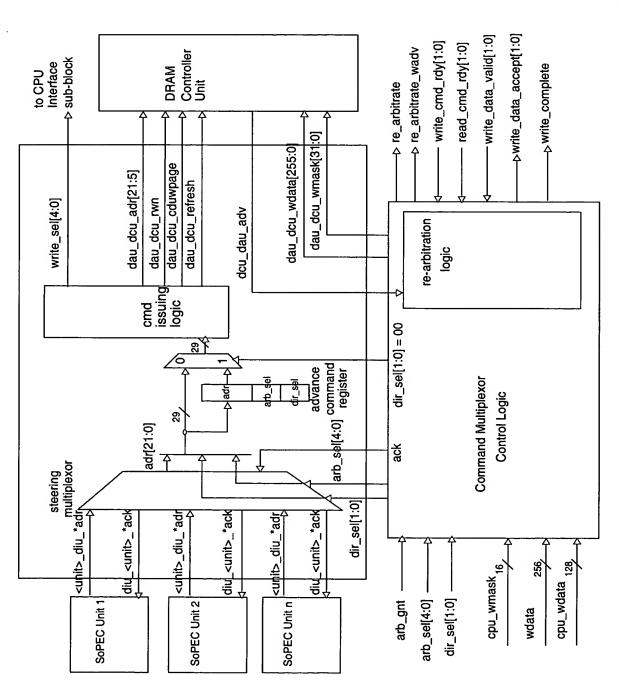


FIG. 114



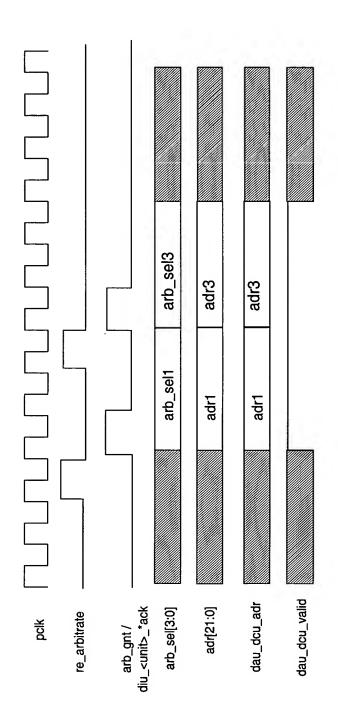


FIG. 116

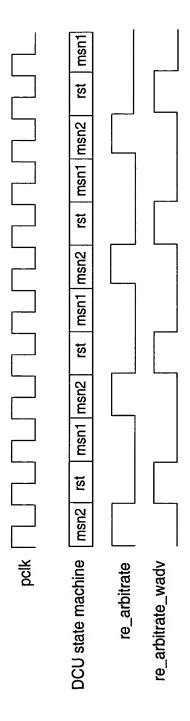


FIG. 117

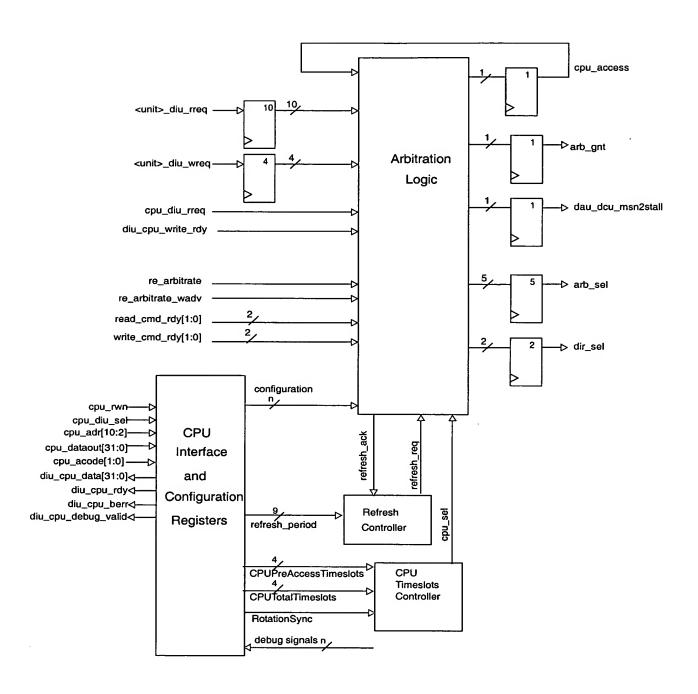


FIG. 118

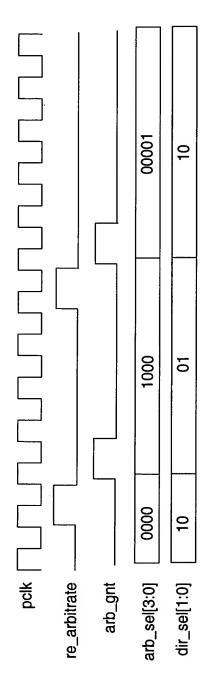


FIG. 119

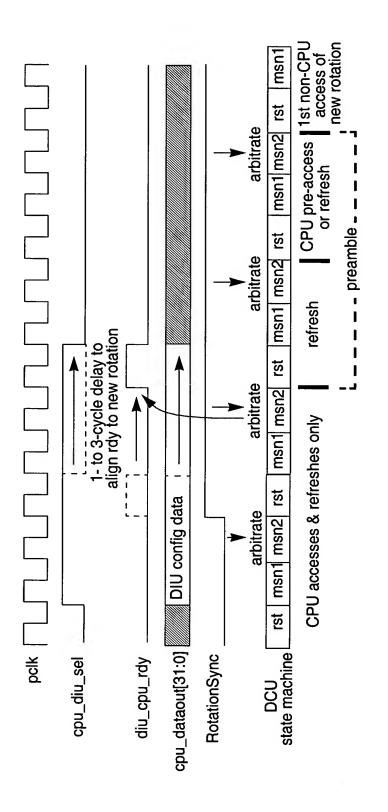


FIG. 120

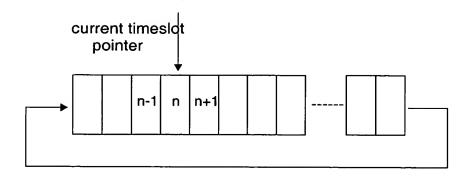


FIG. 121

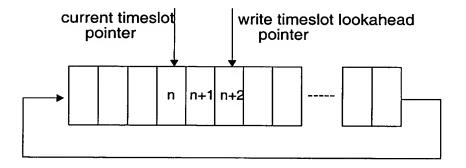


FIG. 122

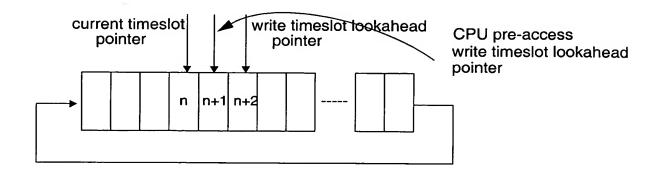


FIG. 123

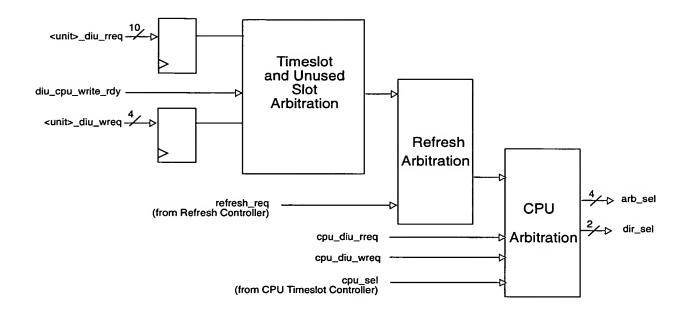


FIG. 124

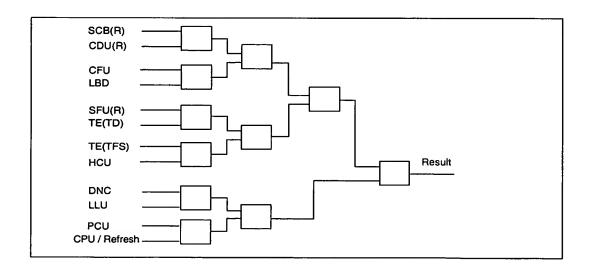
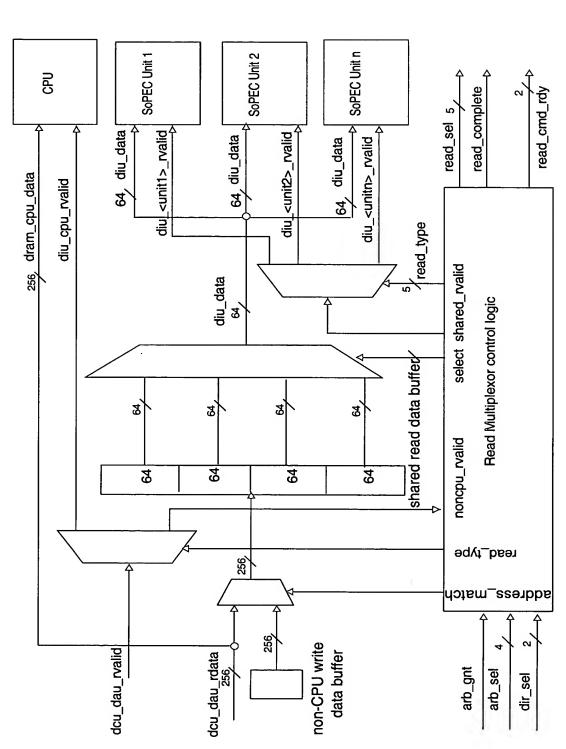


FIG. 125





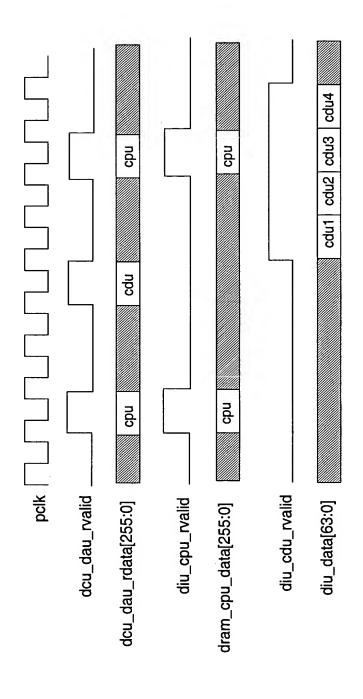


FIG. 127

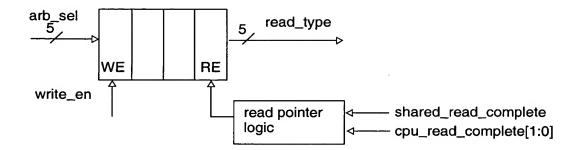


FIG. 128

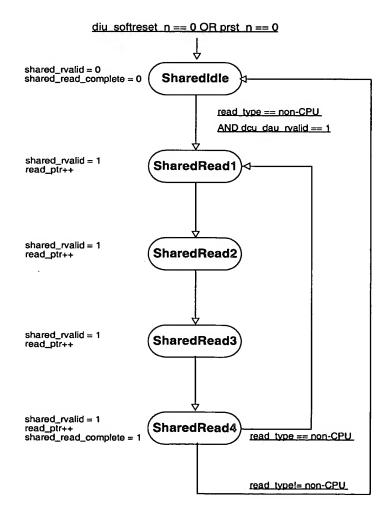


FIG. 129

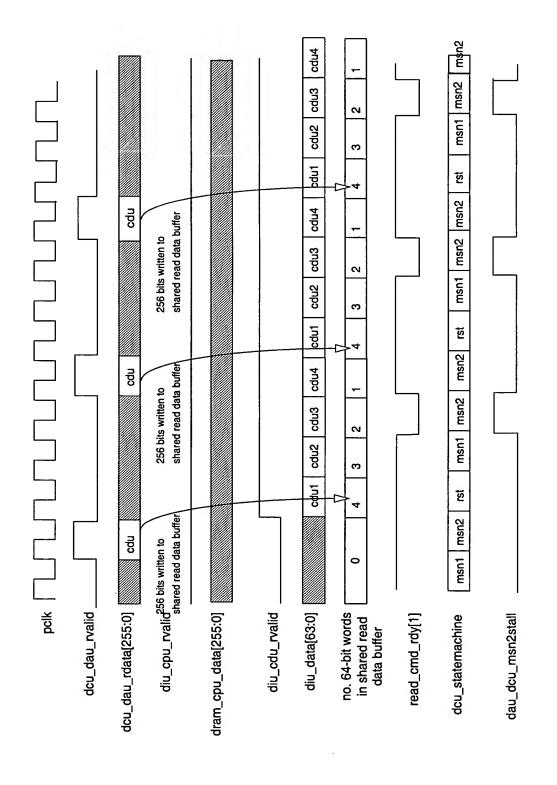


FIG. 130

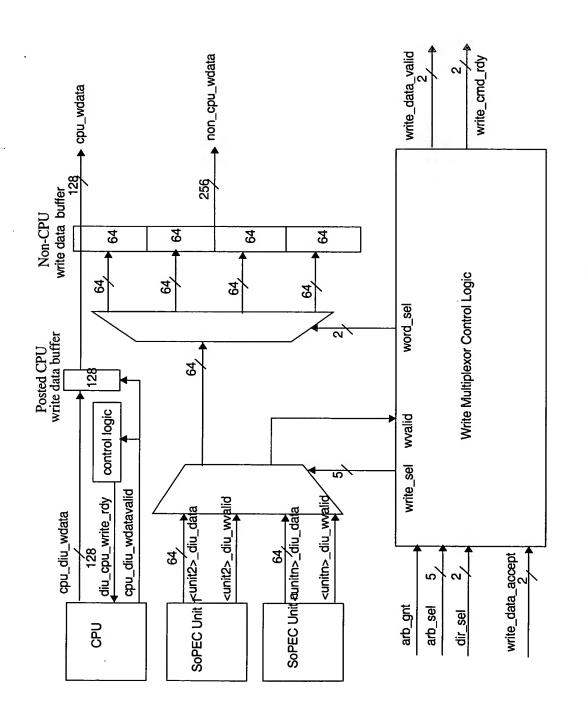


FIG. 131

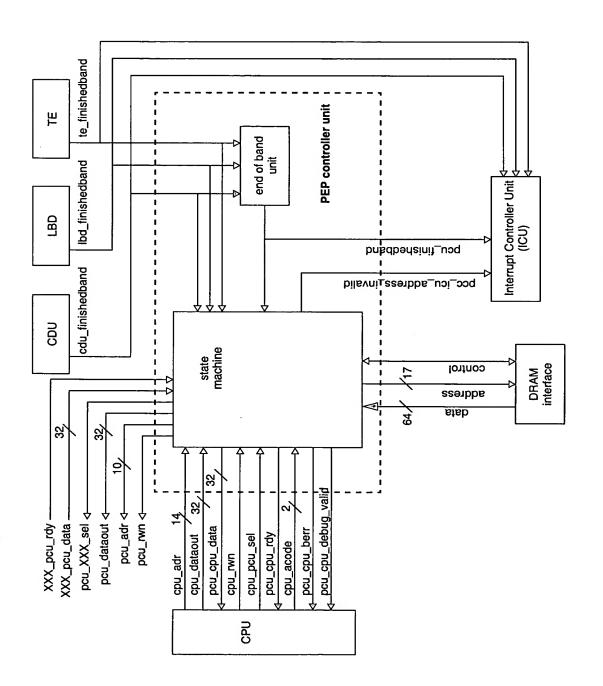


FIG. 132

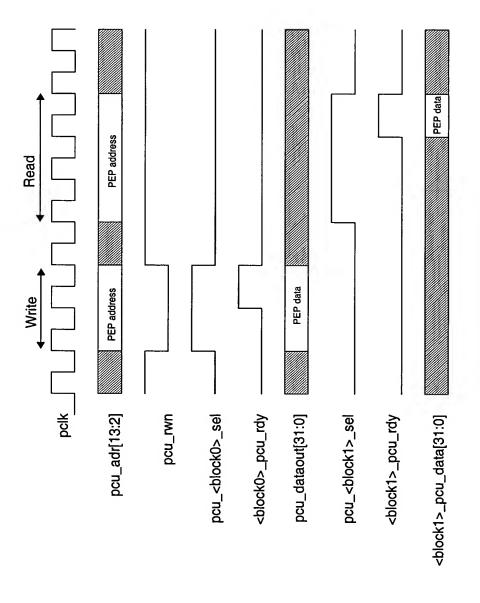


FIG. 133

State Machine A

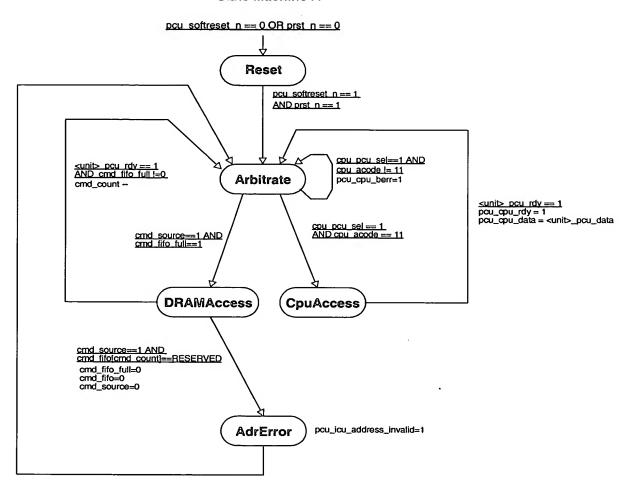


FIG. 134

State Machine B

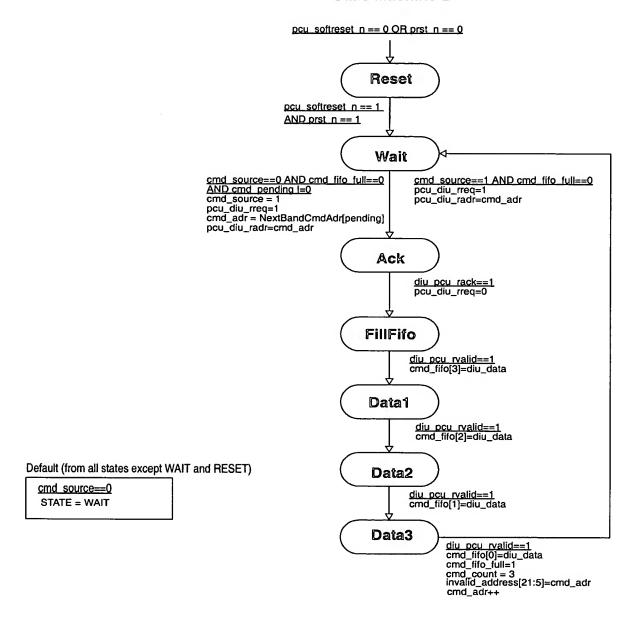


FIG. 135

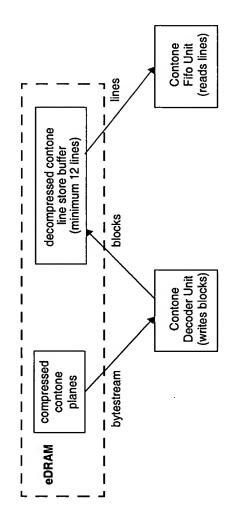


FIG. 136

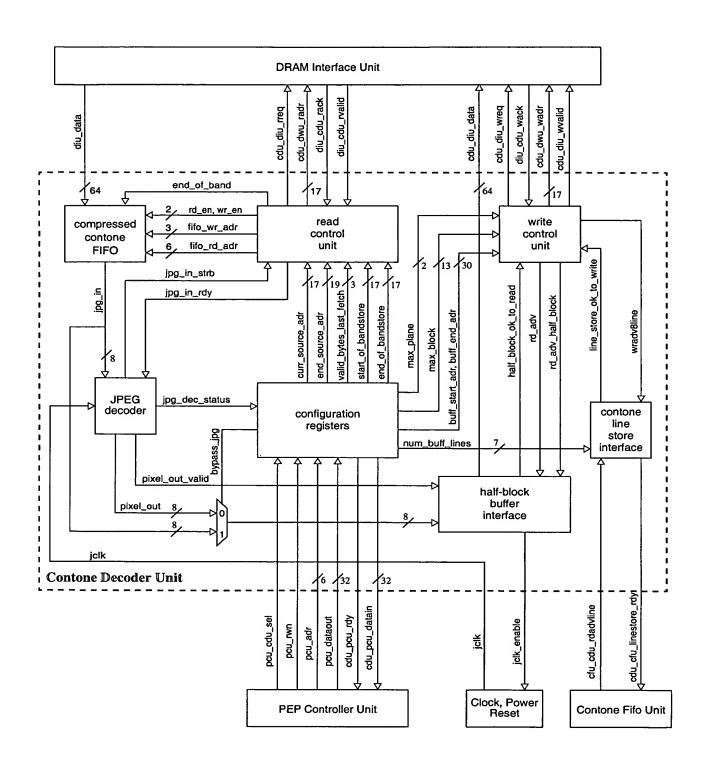


FIG. 137

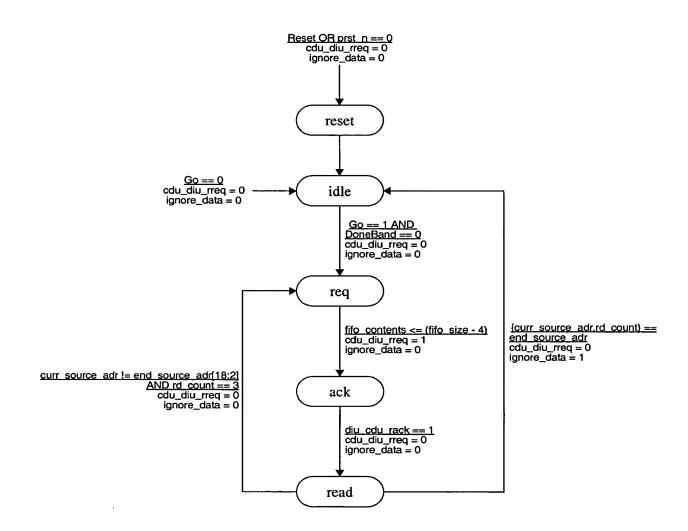


FIG. 138

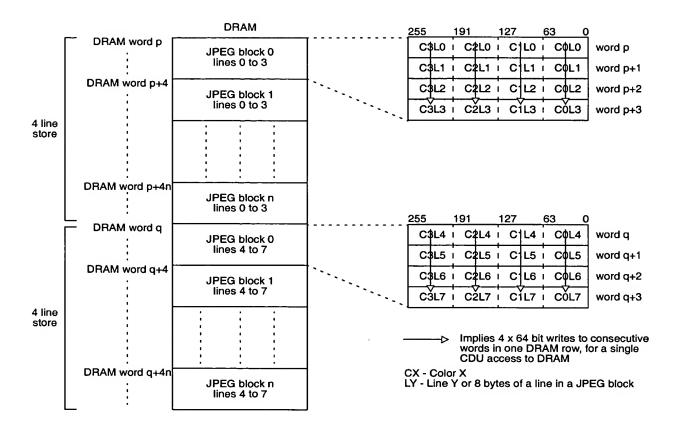
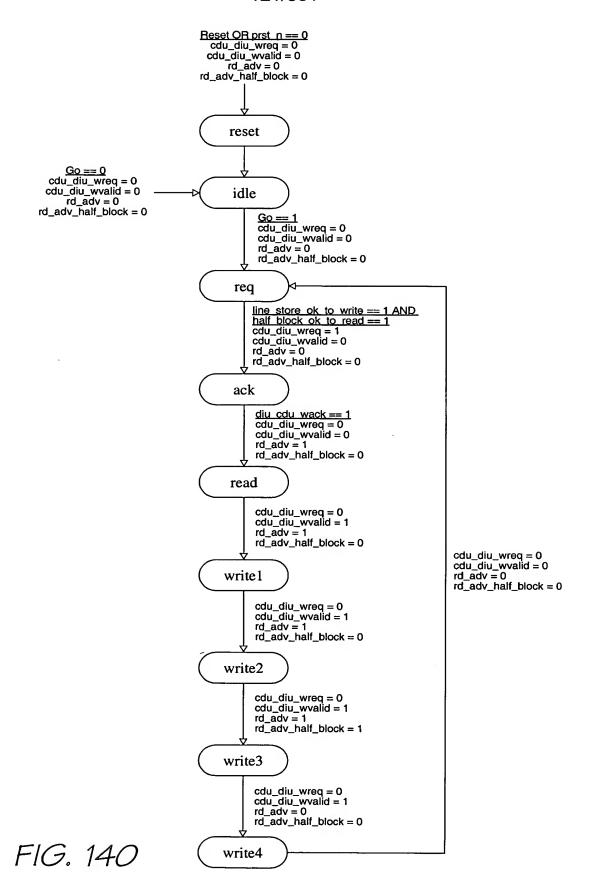


FIG. 139



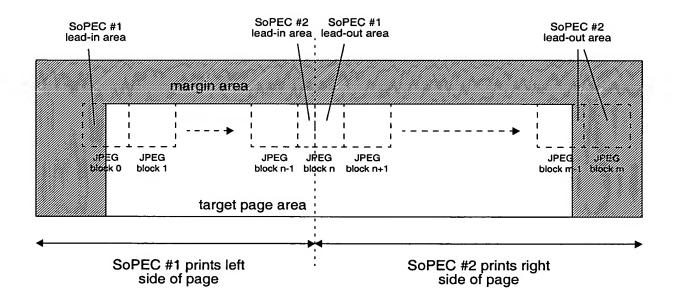


FIG. 141

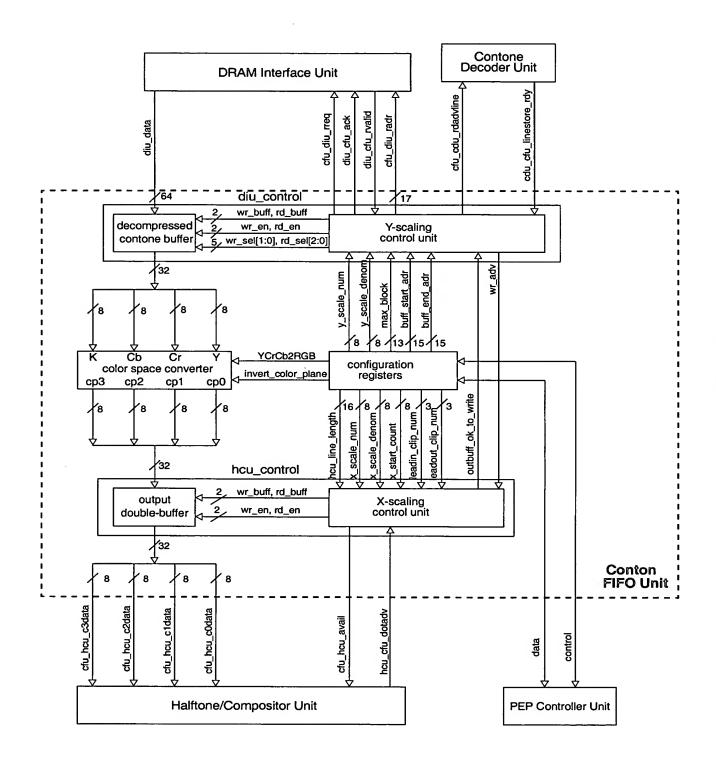


FIG. 142

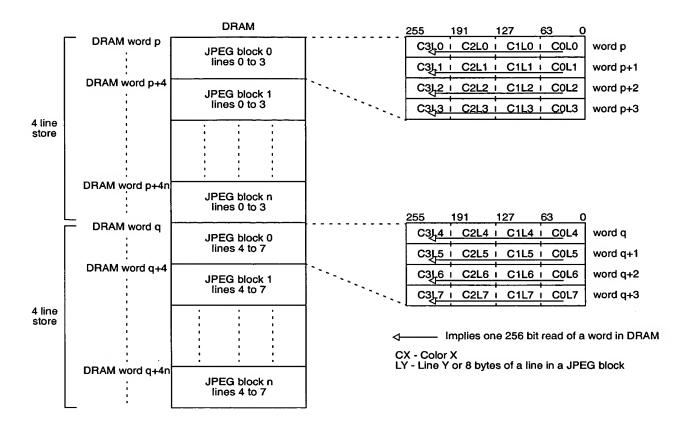


FIG. 143

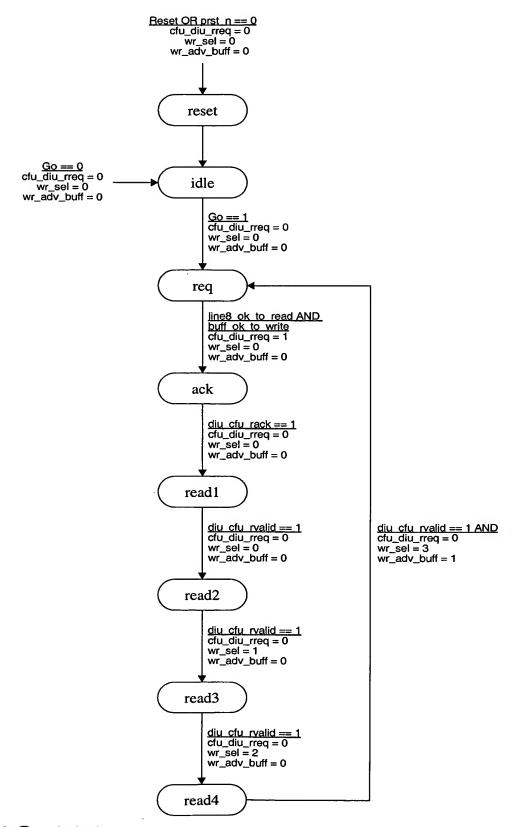


FIG. 144

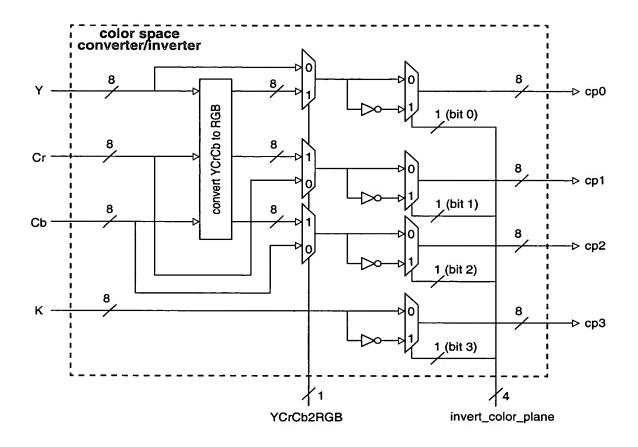


FIG. 145

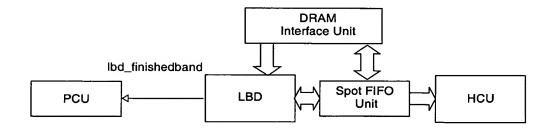


FIG. 146

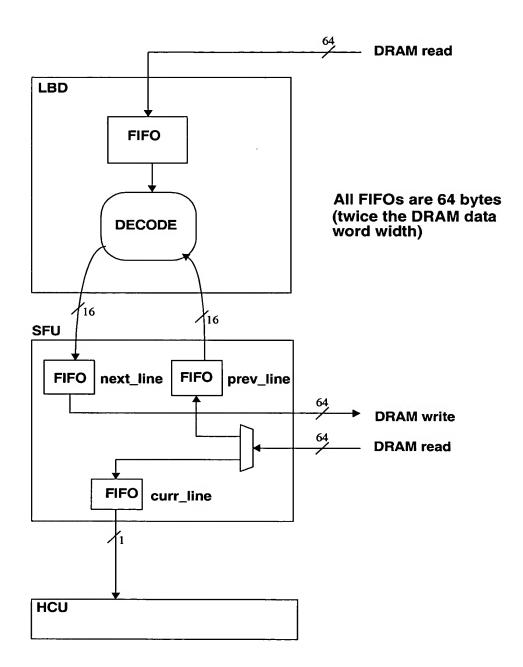


FIG. 147

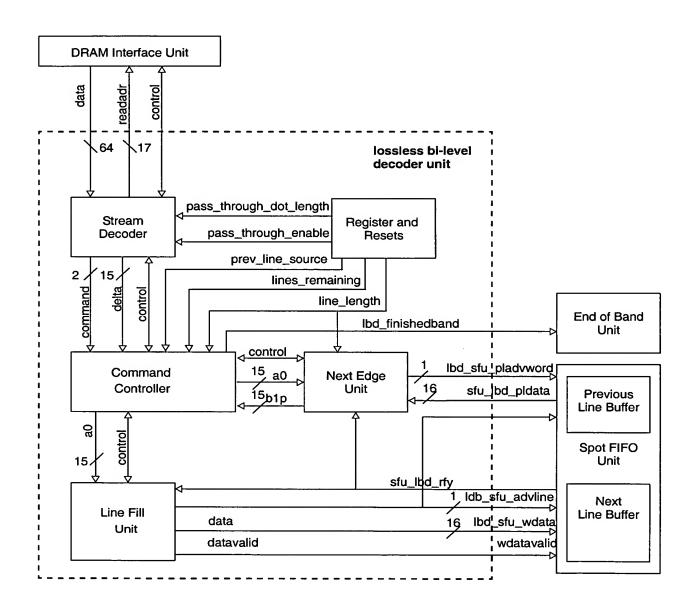


FIG. 148

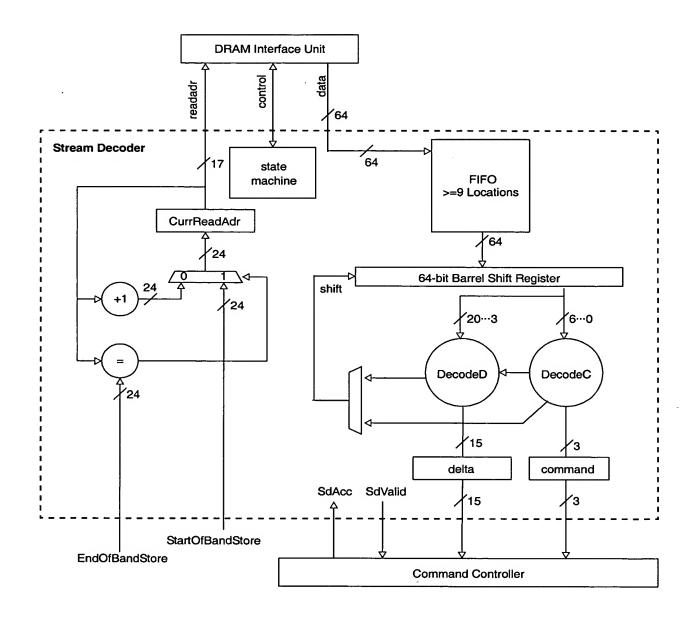


FIG. 149

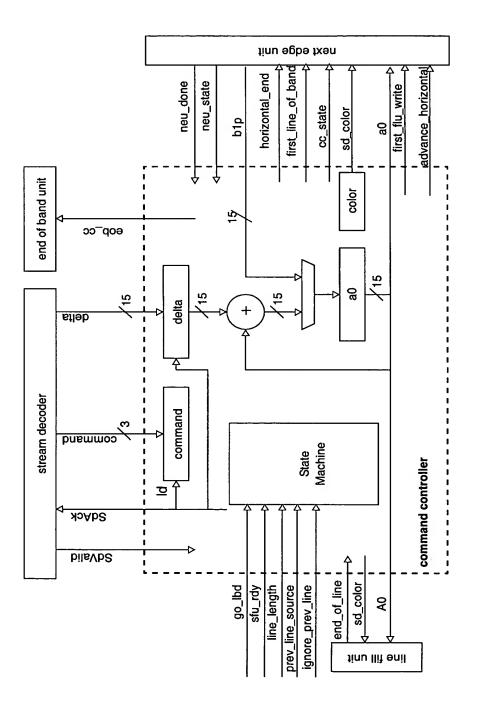


FIG. 150

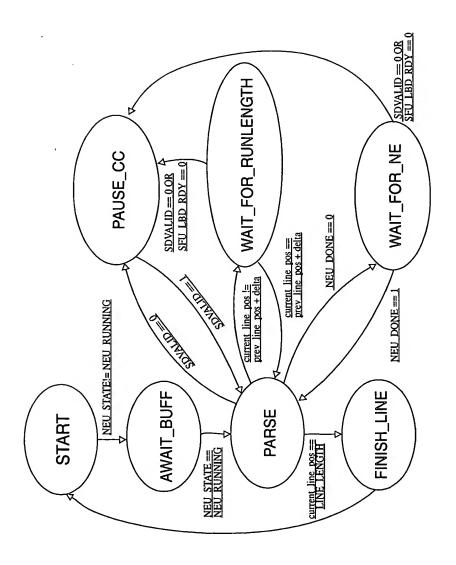


FIG. 151

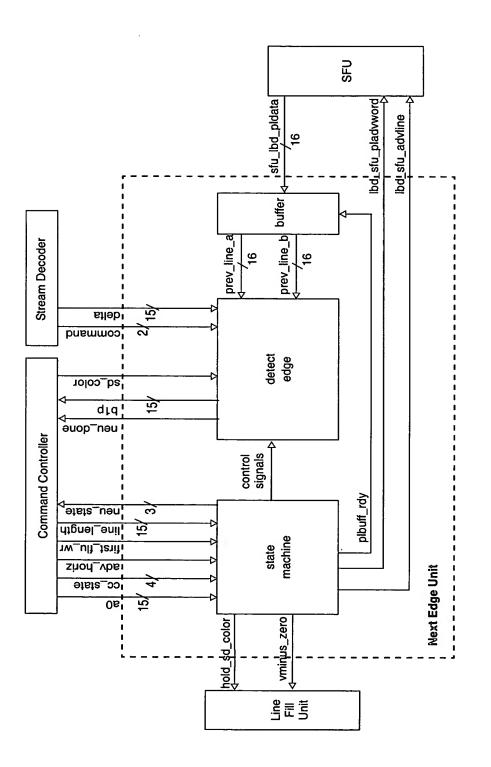


FIG. 152

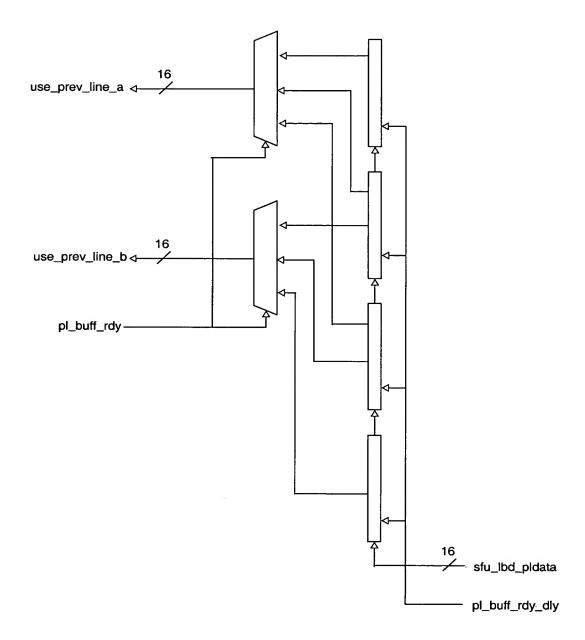


FIG. 153

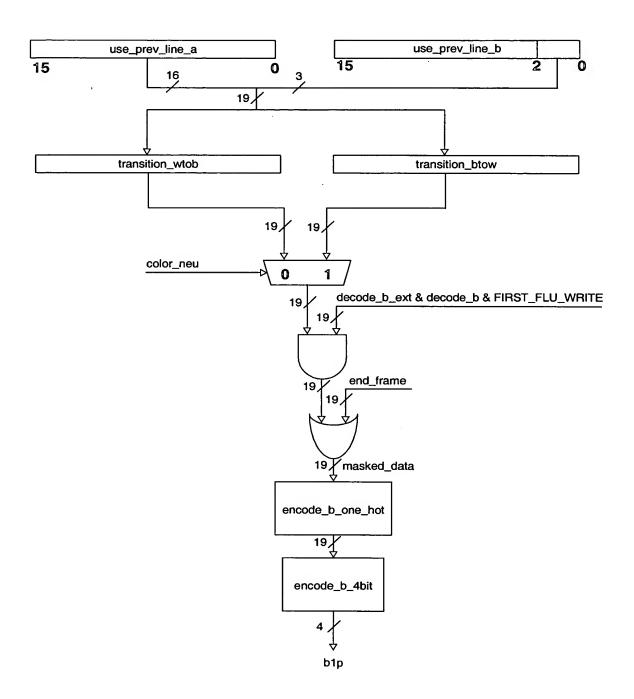


FIG. 154

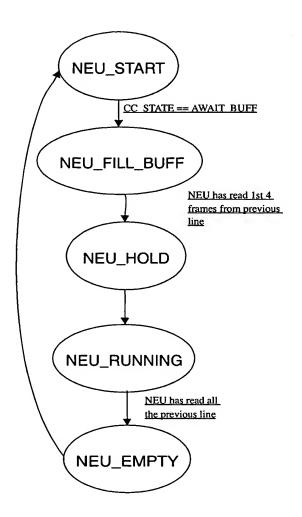


FIG. 155

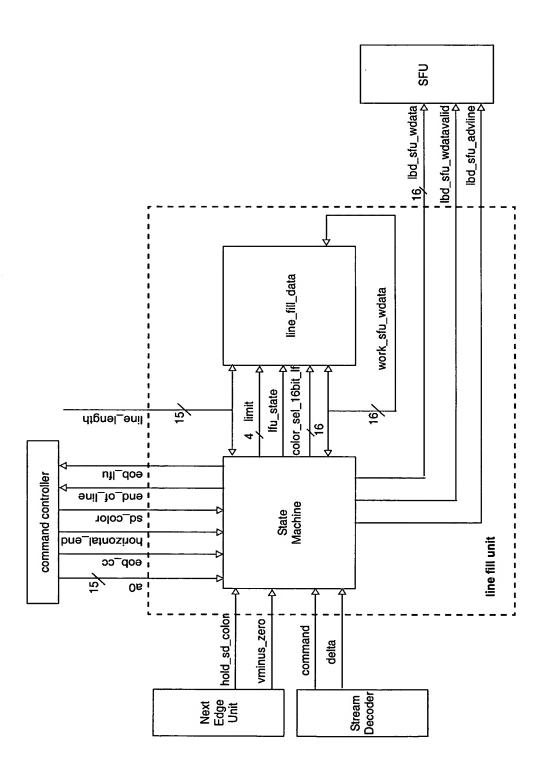


FIG. 156

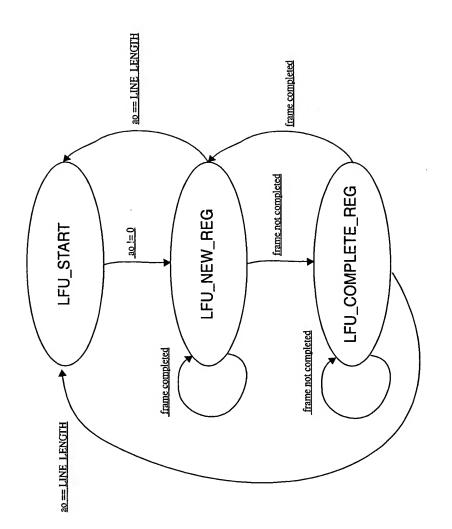


FIG. 157

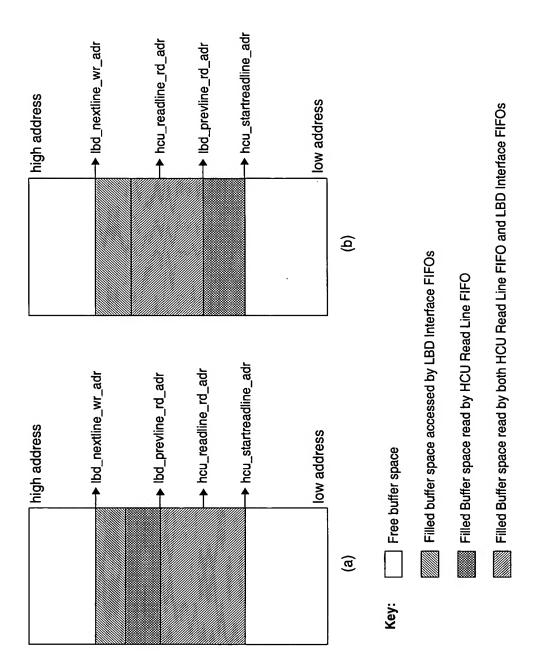


FIG. 158

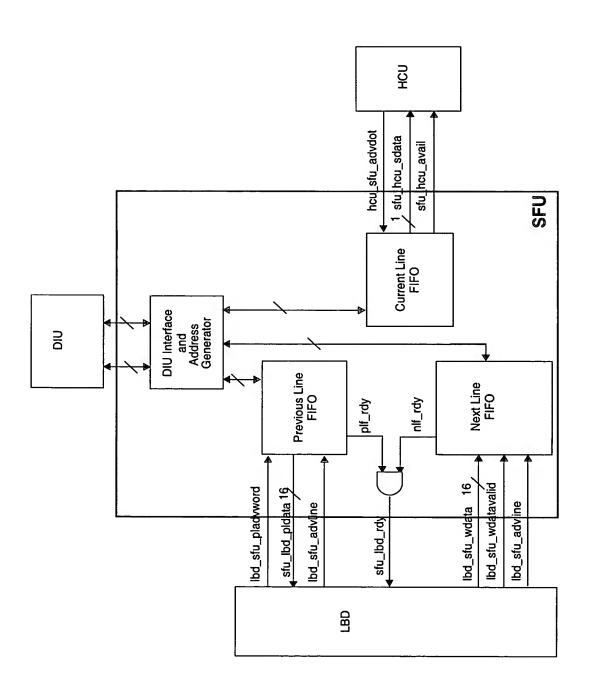


FIG. 159

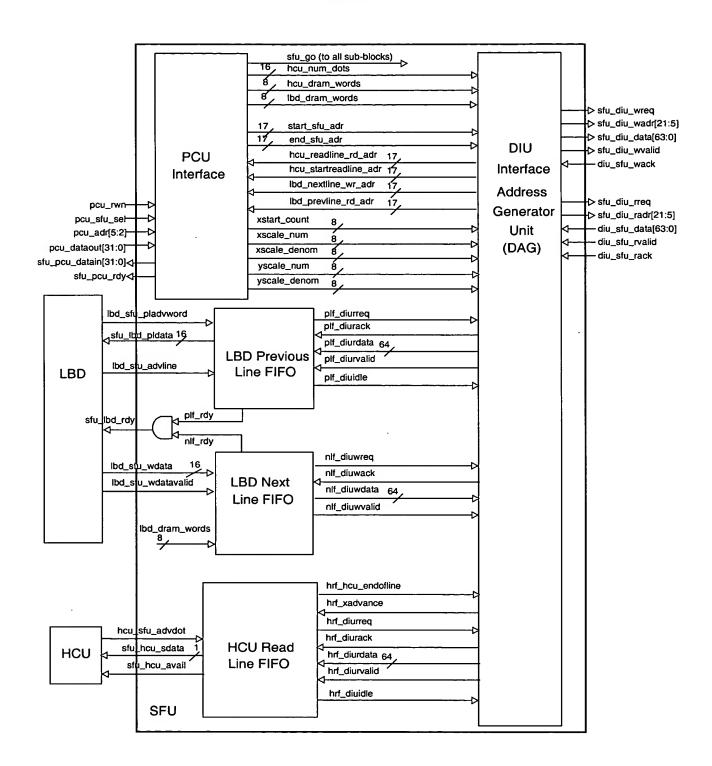


FIG. 160

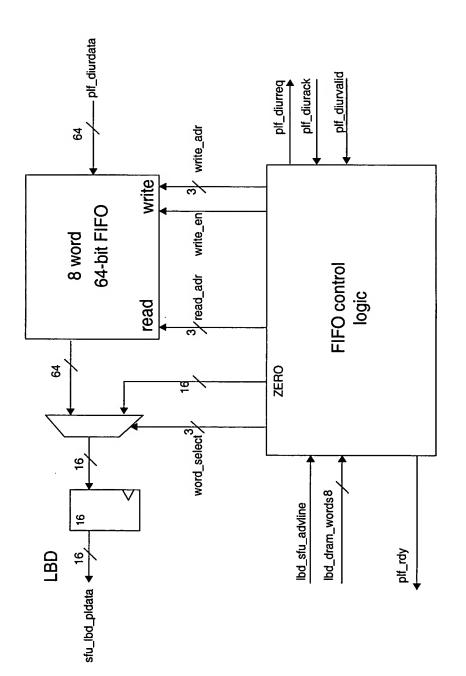


FIG. 161

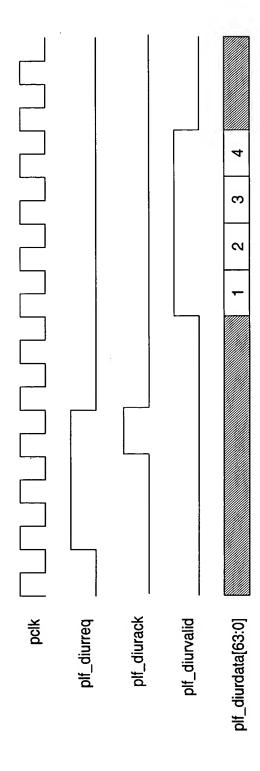


FIG. 162

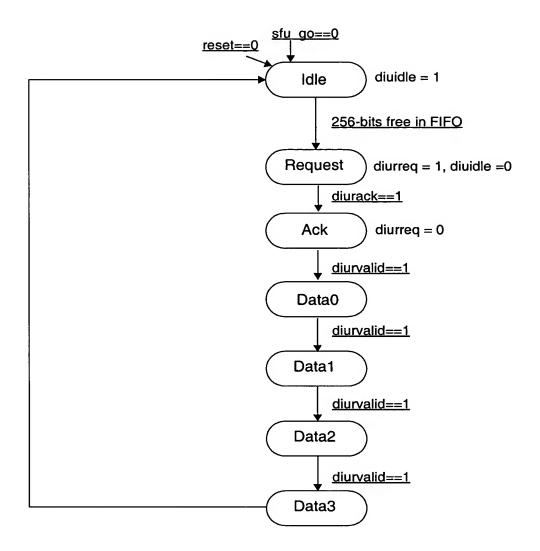


FIG. 163

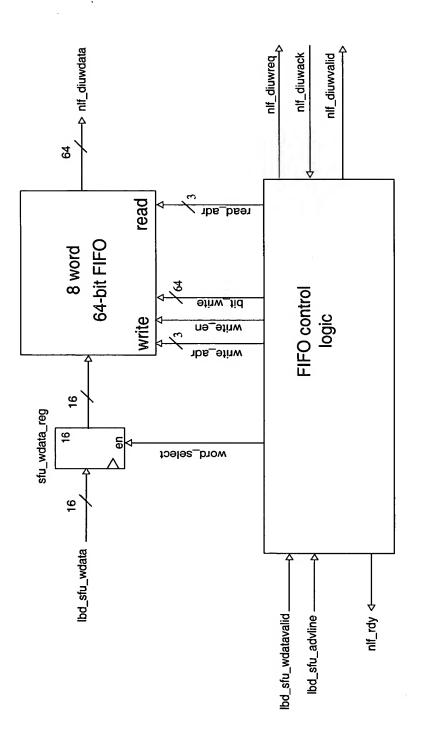
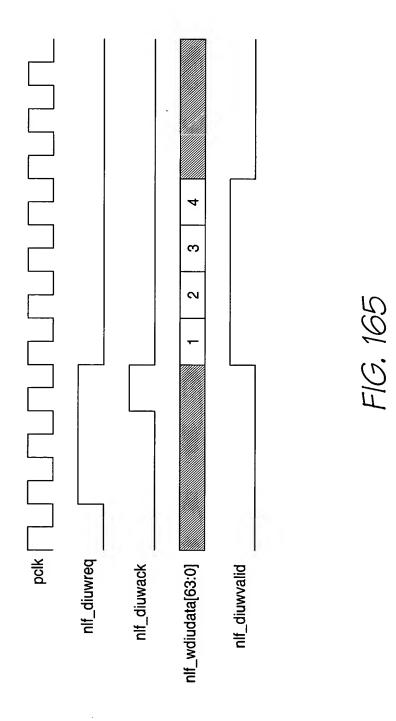


FIG. 164



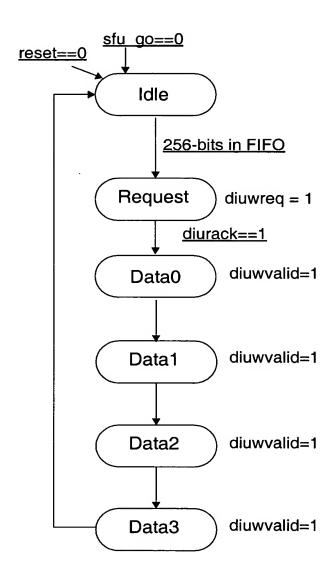


FIG. 166

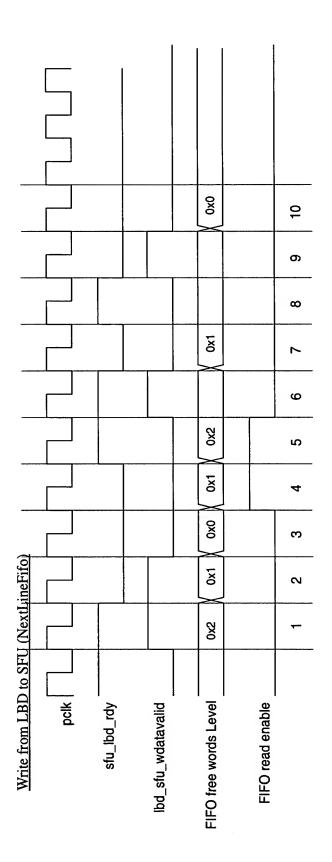


FIG. 167

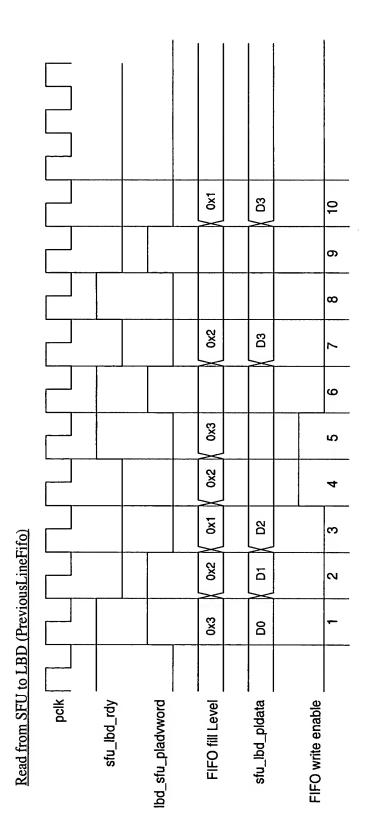


FIG. 168

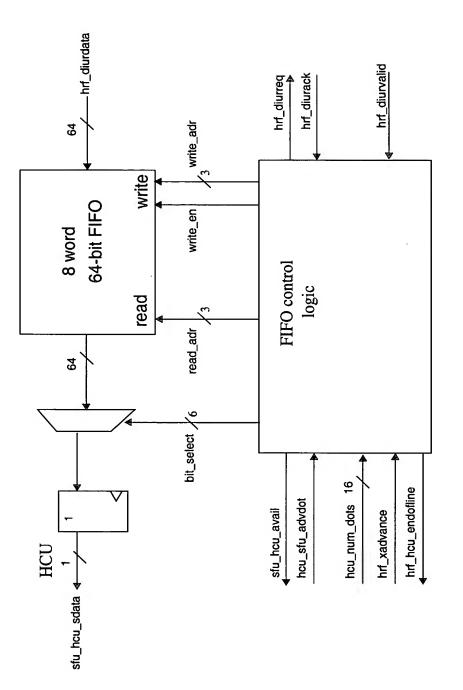


FIG. 169

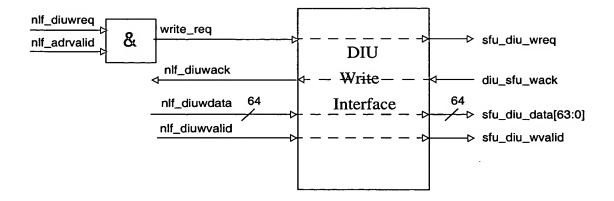


FIG. 170

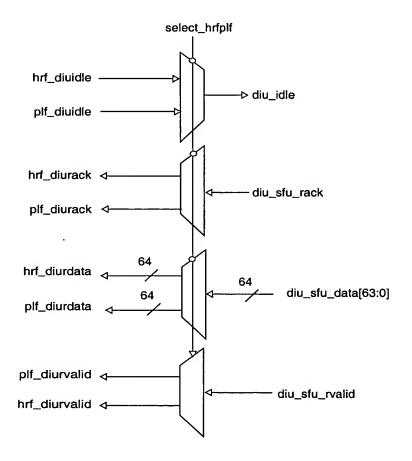


FIG. 171

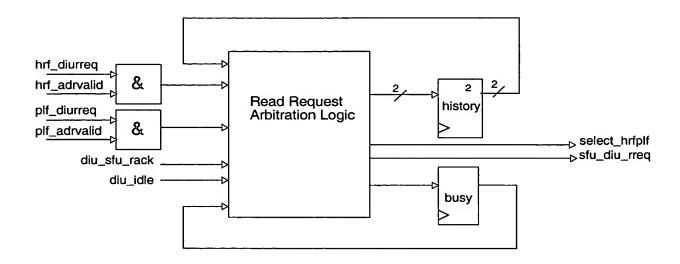


FIG. 172

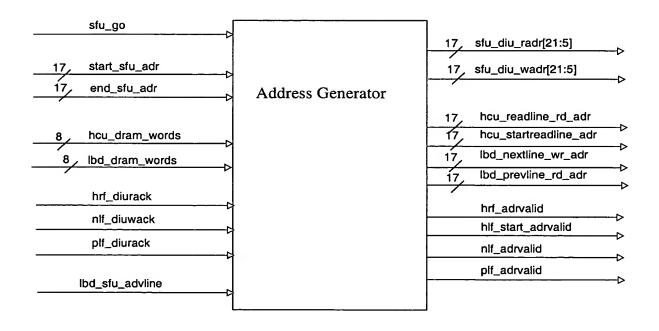


FIG. 173

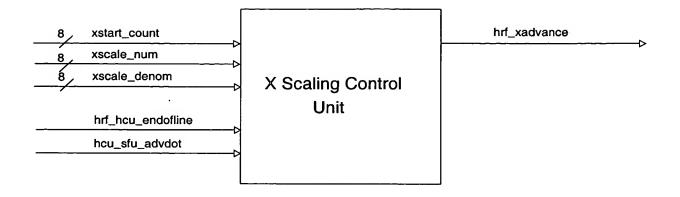


FIG. 174

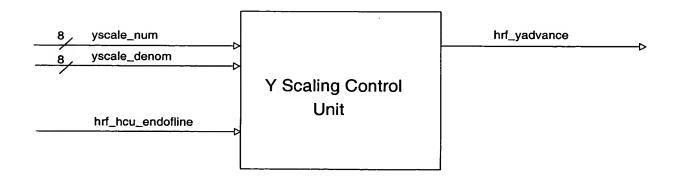


FIG. 175

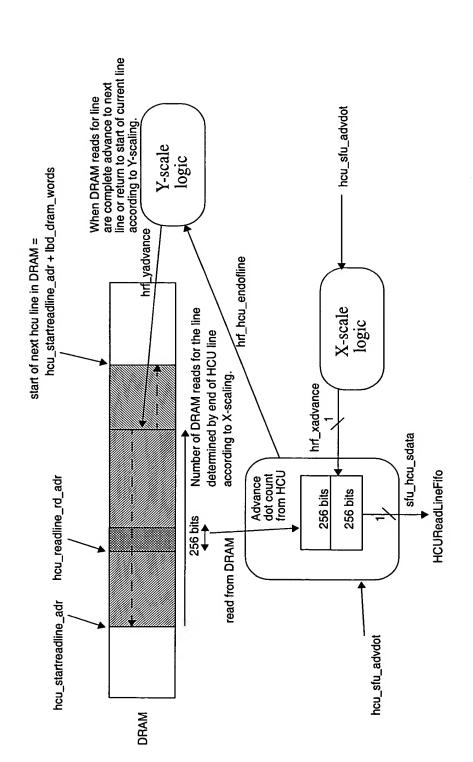


FIG. 176

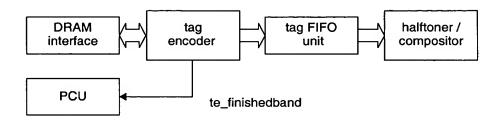


FIG. 177

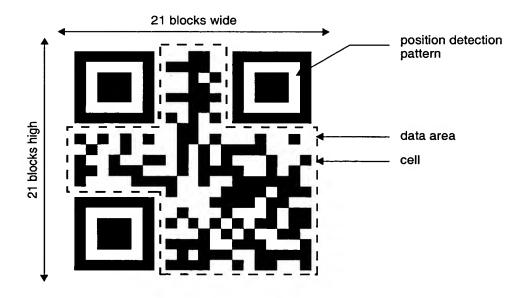
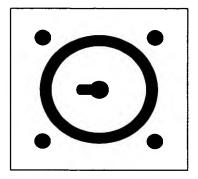
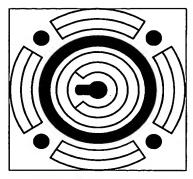


FIG. 178



(a) Netpage tag background pattern



(b) Netpage tag showing data area

FIG. 179

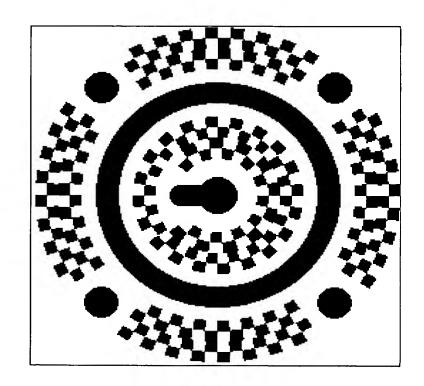


FIG. 180

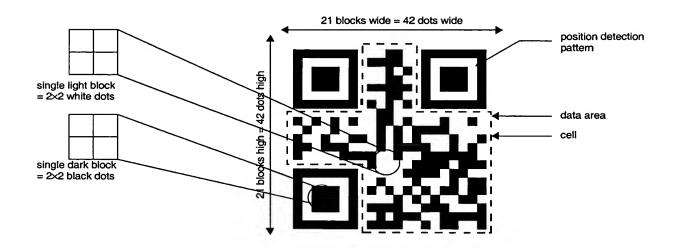


FIG. 181

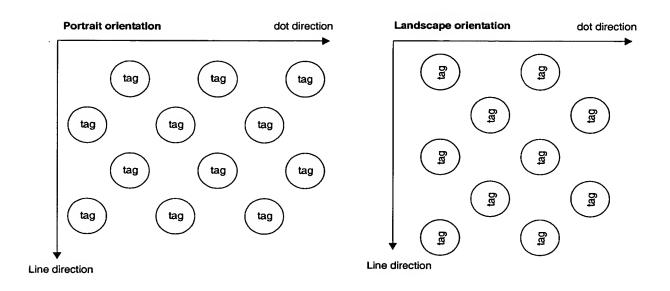


FIG. 182

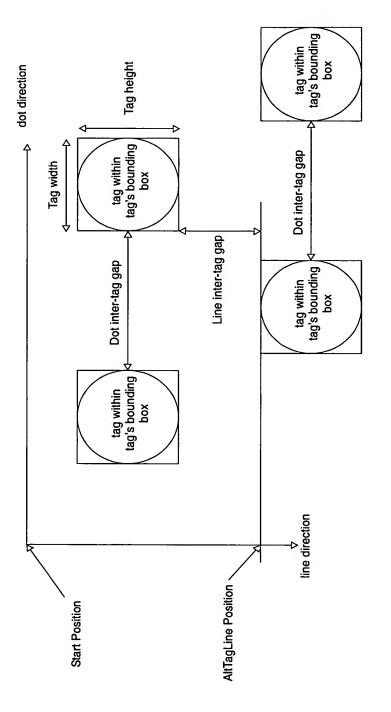
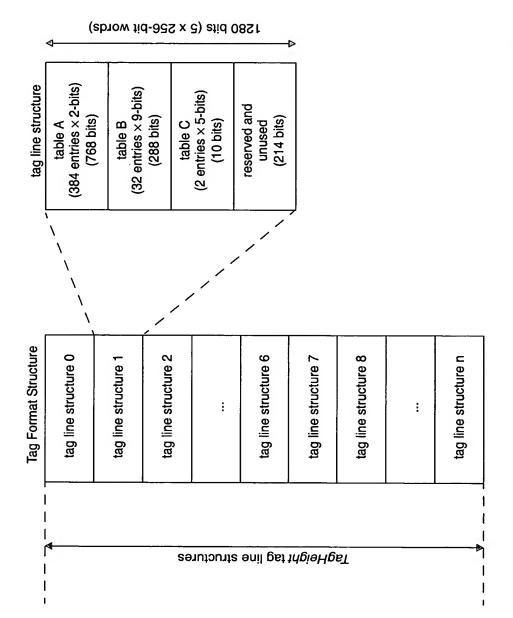
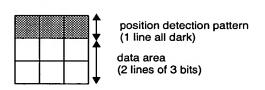


FIG. 183





always 1	always 1	always 1				
(background)	(background)	(background)				
data	data	data				
bit 0	bit 1	bit 2				
data	data	data				
bit 4	bit 5	bit 3				

FIG. 185

Lege	ena
	constant 0
	constant 1
ю	data bit 0
b1	data bit 1
b2	data bit 2
ь3	data bit 3
b4	data bit 4
b5	data bit 5

																				Γ
	ьо	ю	ю	ю	ю			b1	b1	b1	b1	b1			b2	b2	b2	b2	b2	
b0	ьо	b 0	ю	ю	ю	bO	b1	b1	b1	b1	b1	b1	b1	b2	b2	b2	b2	b2	b2	b
ьо	ьо	ю	ьо	ю	ю	ю	b1	b1	b1	b1	b1	b1	b1	b2	b2	b2	b2	b2	b2	b
b 0	ьо	ю	ьо	В	ю	ю	b1	b1	b1	b1	b1	b1	b1	b2						
ьо	ьо	ю	ю	ьо	ю	bO	b1	b1	b1	b1	b1	b1	b1	b2	b2	b2	b2	b2	b2	bź
b0	ьо	ьо	ю	ю	ю	bO	b1	ь1	b1	b1	b1	b1	b1	b2	b2	b2	b2	b2	b2	bź
	bO	bO	ю	ю	ю			b1	ь1	b1	b1	b1			b2	b2	b2	b2	b2	
	b4	b4	b4	b4	b4			b5	b5	b5	b5	b5			ьз	ಜ	b3	b3	ьз	
b4	b4	b4	b 4	b 4	b 4	b 4	b5	b5	b5	b5	b5	b5	b5	В	ьз	ьз	b3	ь3	b3	ь
b4	b4	b4	b 4	b4	b 4	b 4	b5	b 5	b5	b5	b5	b5	b 5	ьз	ьз	ьз	b3	b3	b3	ь
b4	b4	b4	b4	b4	b4	Ь4	b5	b5	b5	b5	b5	b5	b5	ьз	ьз	ьз	b3	ьз	ьз	ы
b4	b4	b4	b4	b4	b4	b4	b5	b5	b5	b 5	b5	b5	b5	b3	ьз	ьз	b3	ьз	ьз	bí
b4	b4	b4	b4	b4	b4	b 4	b5	b5	b5	b5	b5	b5	b5	b3	b3	ьз	b3	ьз	ьз	ь
	b4	b4	b4	b4	b4			b 5	b5	b5	b5	b5			ьз	ьз	b3	ьз	bЗ	

FIG. 186

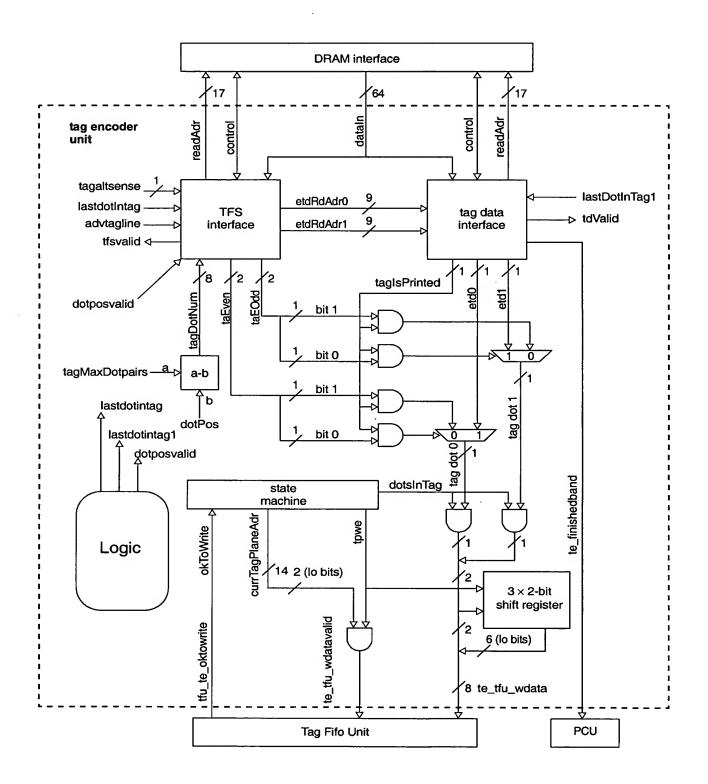


FIG. 187

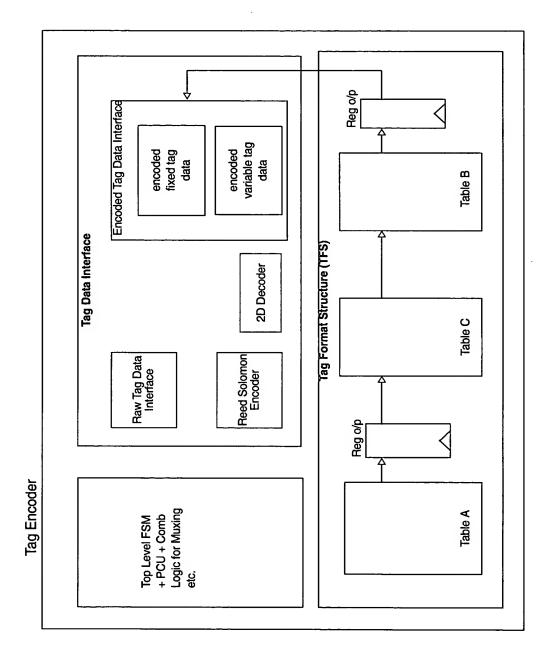


FIG. 188

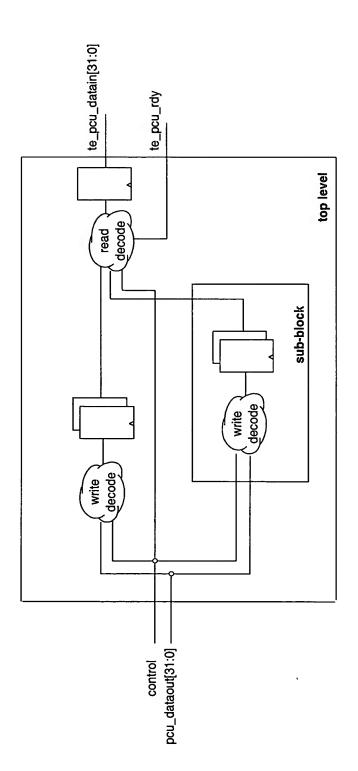


FIG. 189

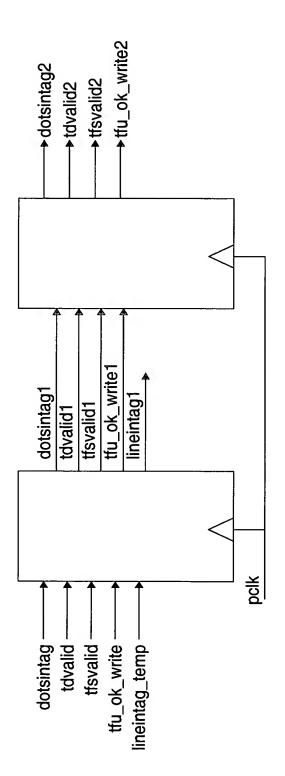


FIG. 191

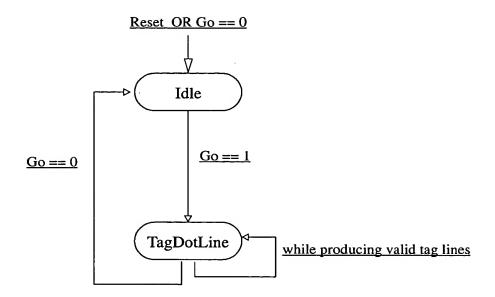


FIG. 190

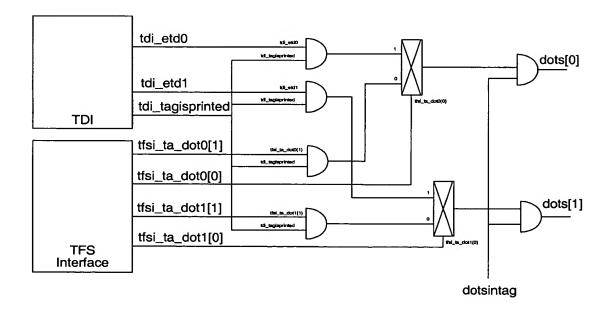


FIG. 192

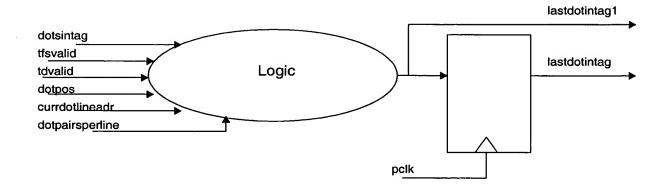


FIG. 193

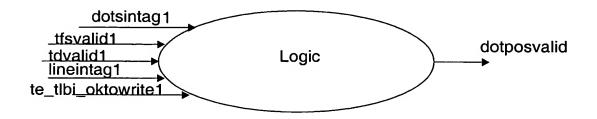


FIG. 194

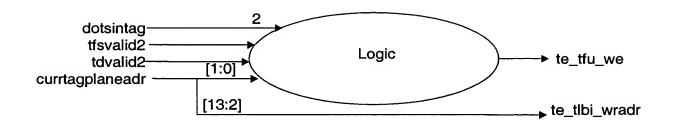


FIG. 195

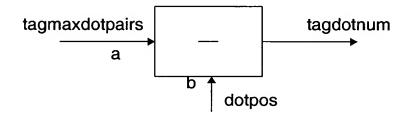


FIG. 196

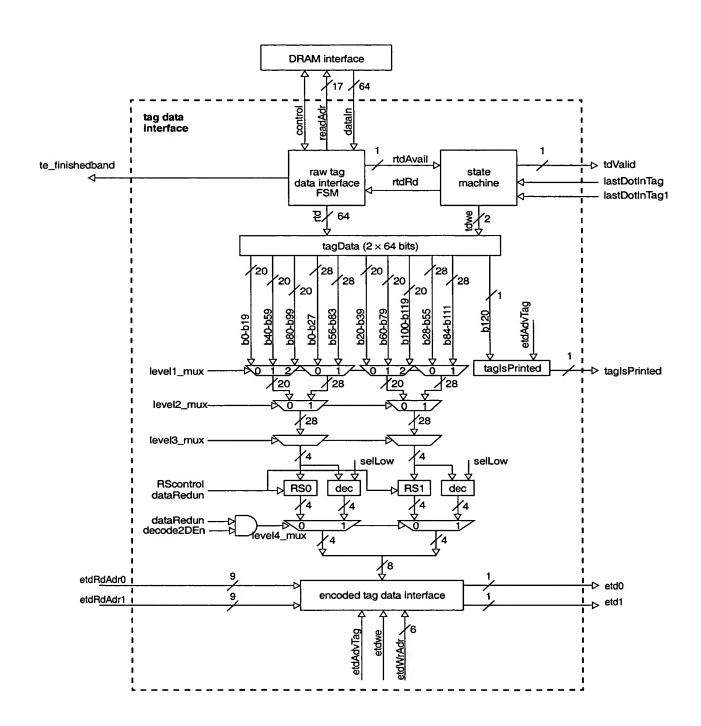


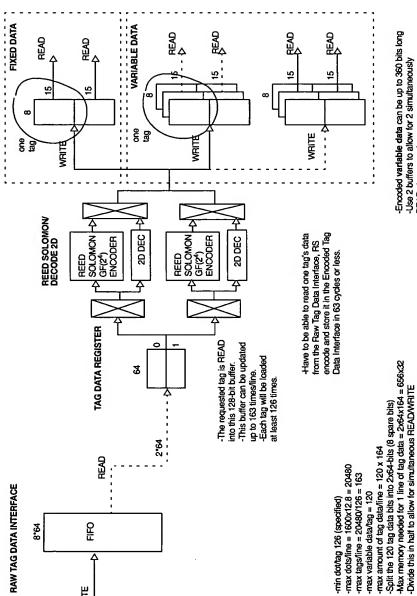
FIG. 197

ENCODED TAG DATA INTERFACE

Encoded fixed data can be up to 120 bits long Use 2 buffers to allow for 2 simultaneously

READs in one cycle.

These stores hold the fixed tag data for 1 tag. -Total memory = 120x2 = 240 bits



WRITE

Encoded variable data can be up to 360 bits long -Use 2 buffers to allow for 2 simultaneously READs in one cycle.

READWRITE

Once printing has started each half buffer has 1/2 a line in which to be loaded i.e. for a 12,8 inch line it has 10240 dots or 5120 cycles for an 8 inch line it has 6400 dots or 3200 cycles

-Therefore the data will be updated at most every 1290240 cycles.

-Total memory = 164x2x64 = 20992-bits

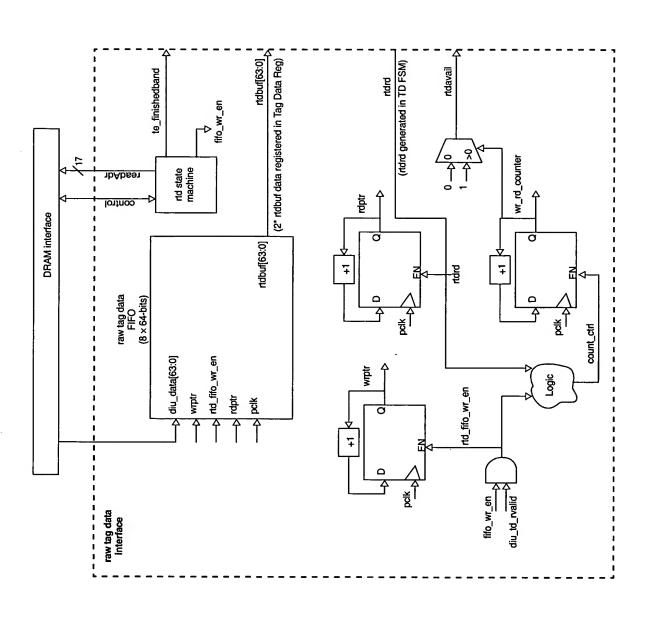
-The store uses 9-bit addressing. Bit-9 indicates which buffer.

-From the specification, we must be able to process 2 dots/cycle.

-126 lines contains 20480x126 = 2580480 dots.

Once all this data is loaded it will be valid for at least 126 lines.

-Total memory = 360x2x2 = 1280 bits
-Min tag width = 126 dots
so the fastlest that 1 tag can be read = 126/2 = 63 cycles -Use 2 buffers to allow for simultaneously



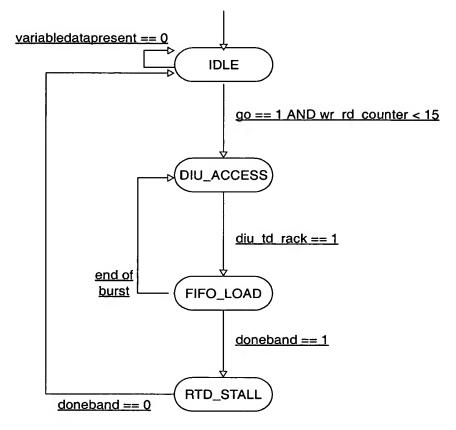


FIG. 200

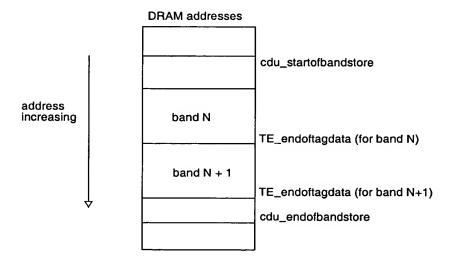
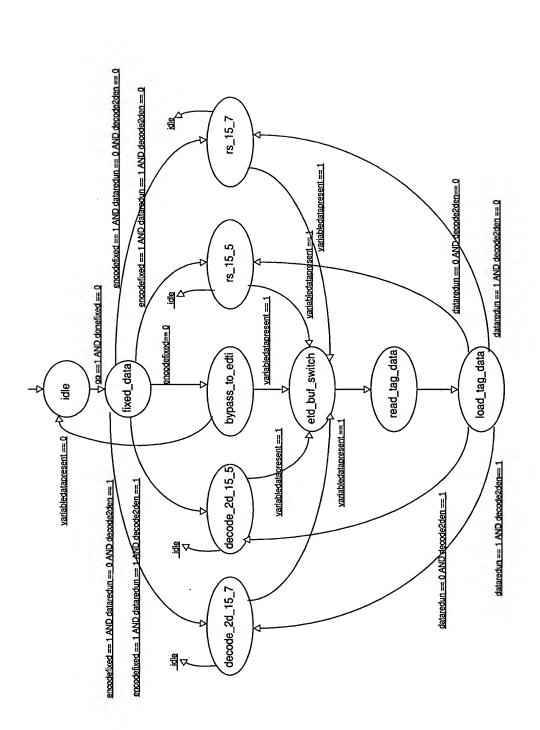


FIG. 201



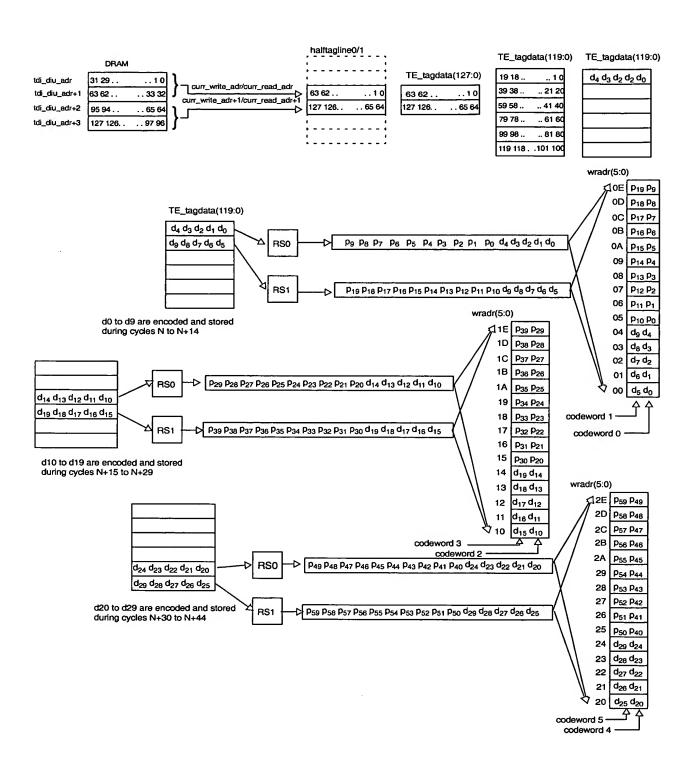


FIG. 203

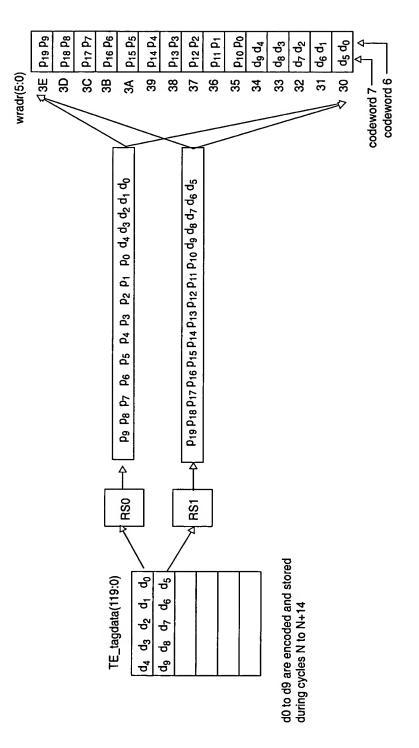


FIG. 204

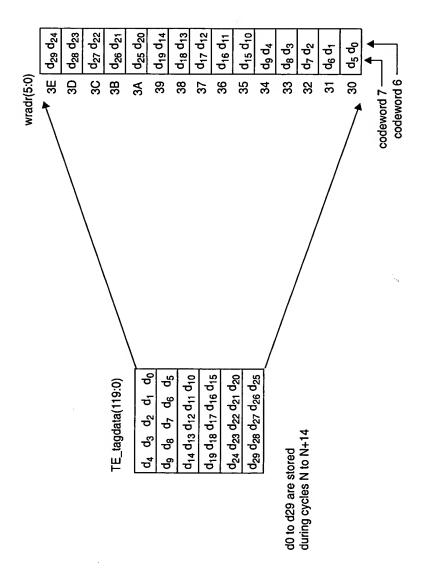


FIG. 205

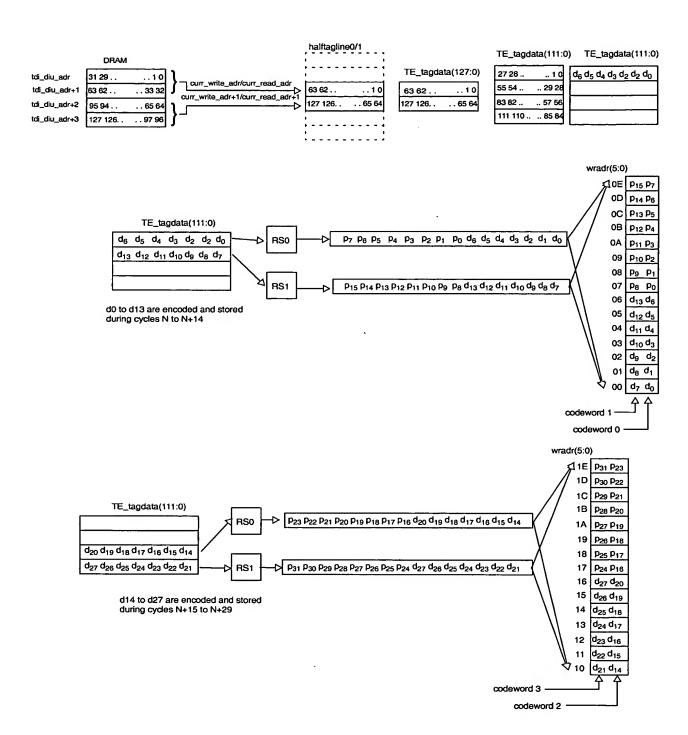


FIG. 206

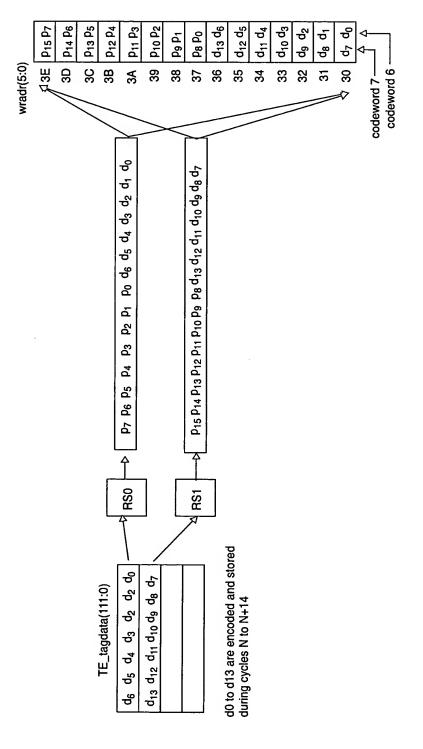


FIG. 207

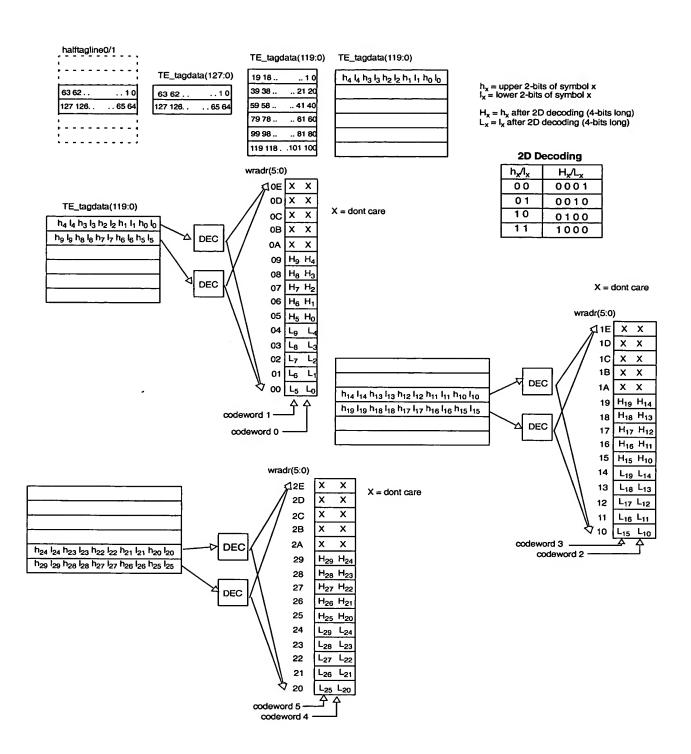
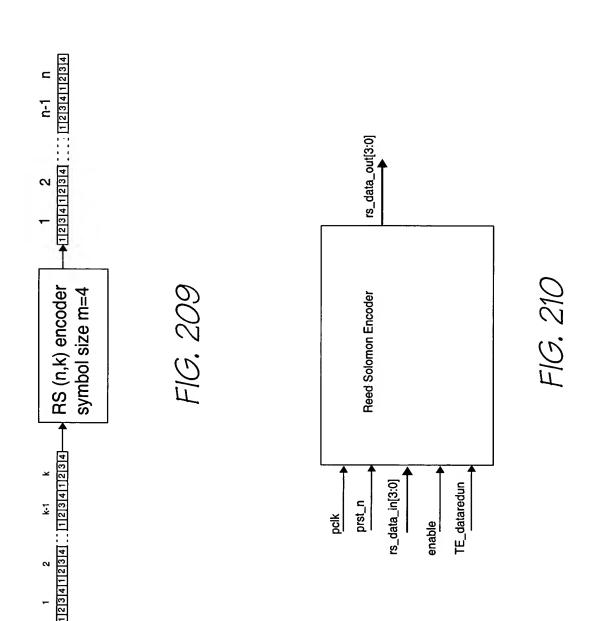


FIG. 208



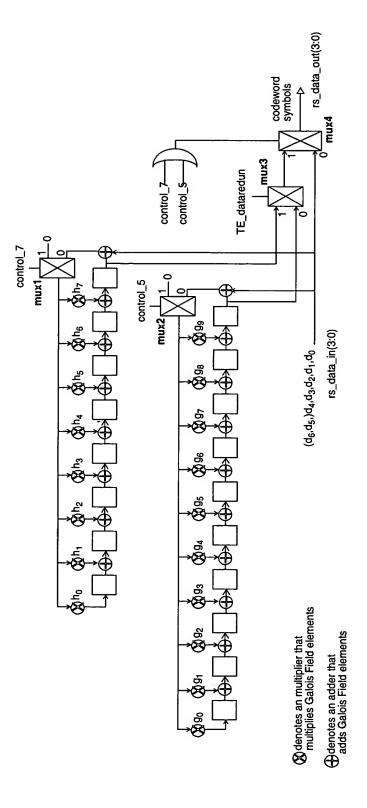


FIG. 211

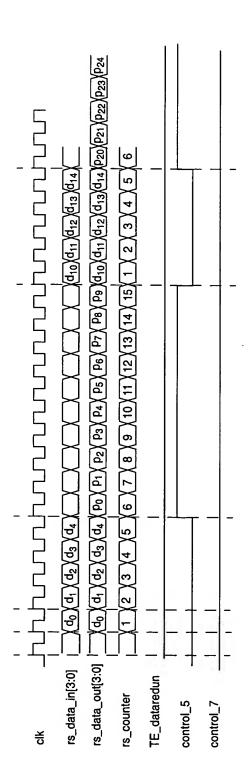


FIG. 212

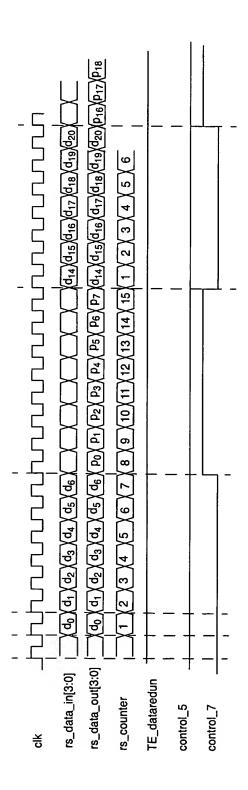
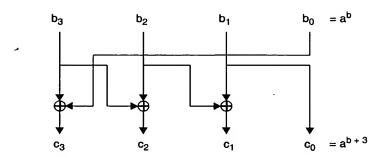
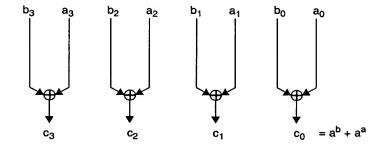


FIG. 213



exclusive OR gate

FIG. 214



exclusive OR gate

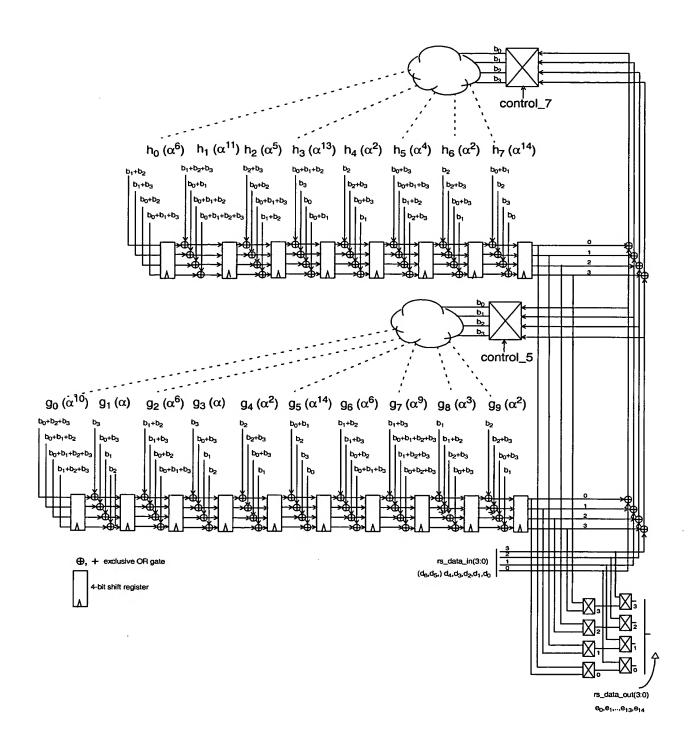


FIG. 216

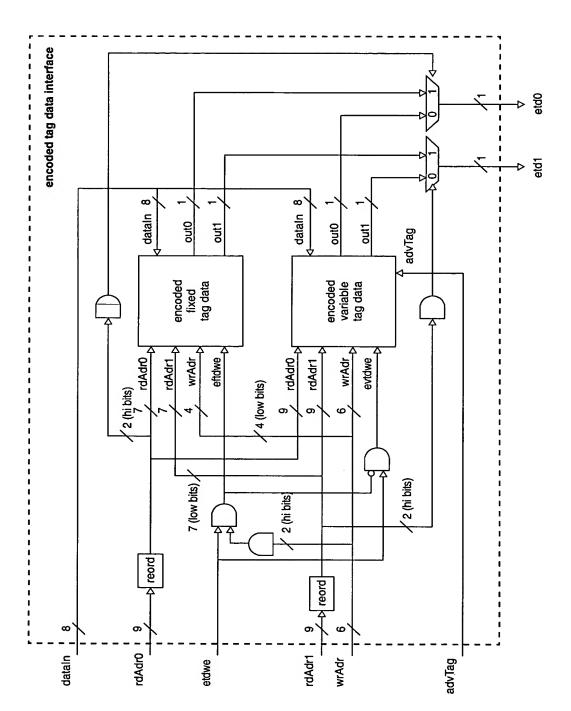


FIG. 217

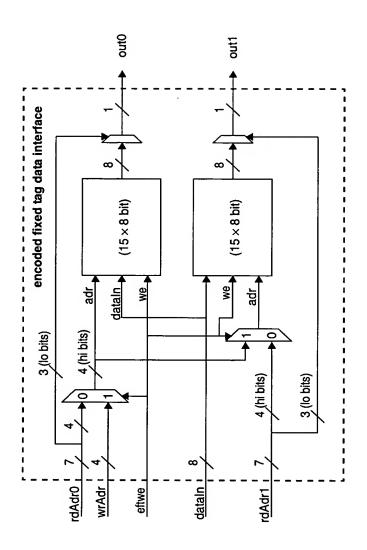


FIG. 218

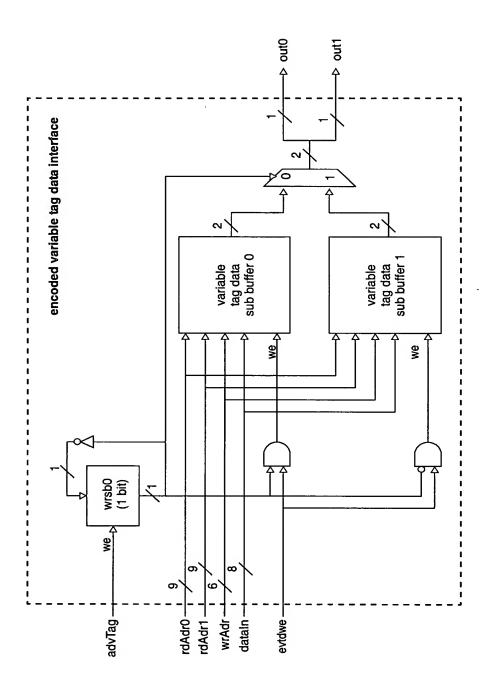


FIG. 219

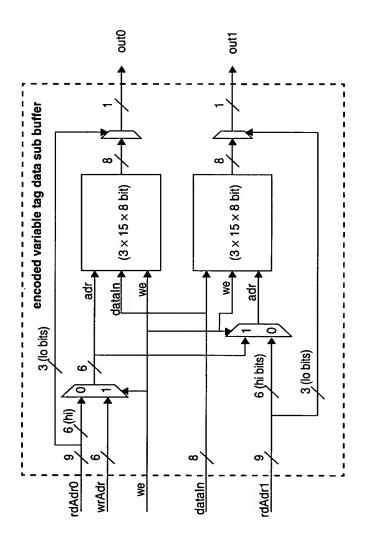


FIG. 220

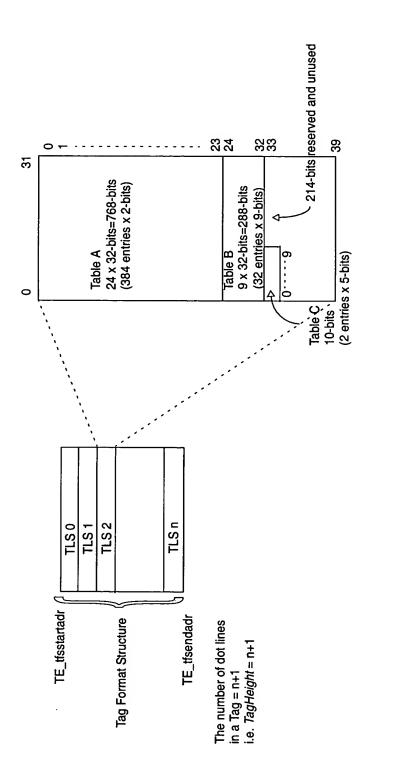


FIG. 221

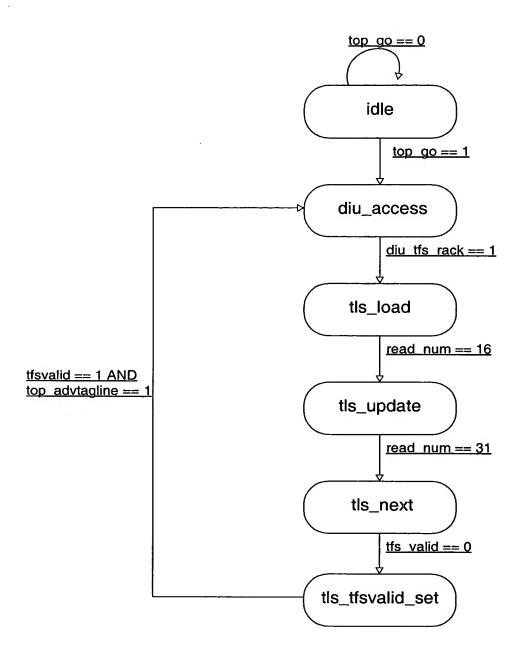


FIG. 222

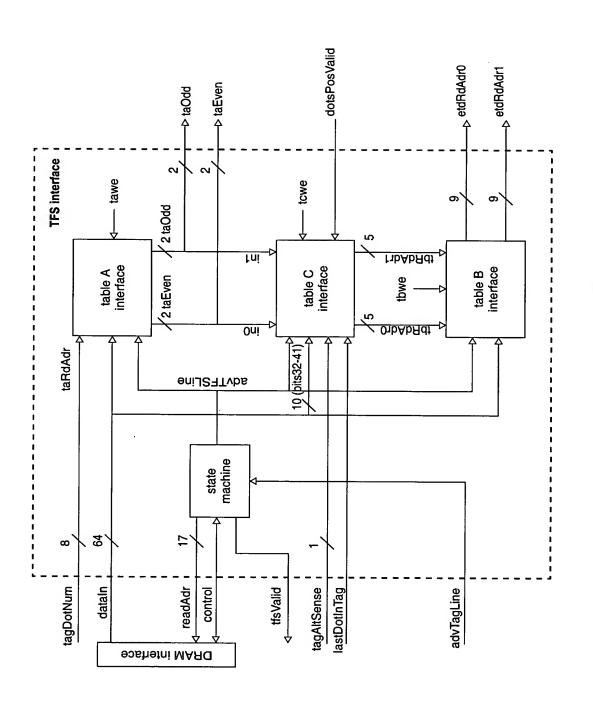


FIG. 223

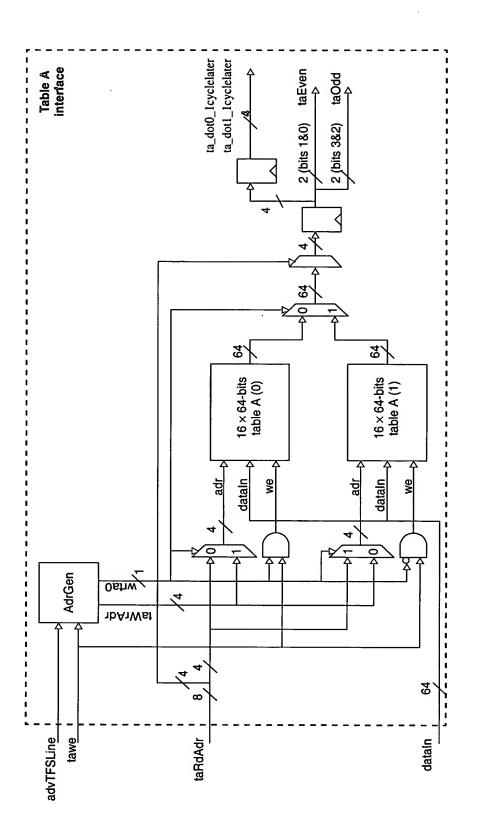


FIG. 224

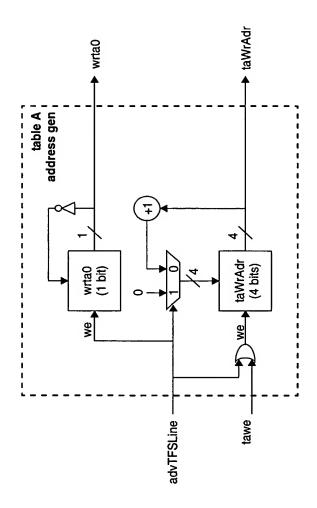


FIG. 225

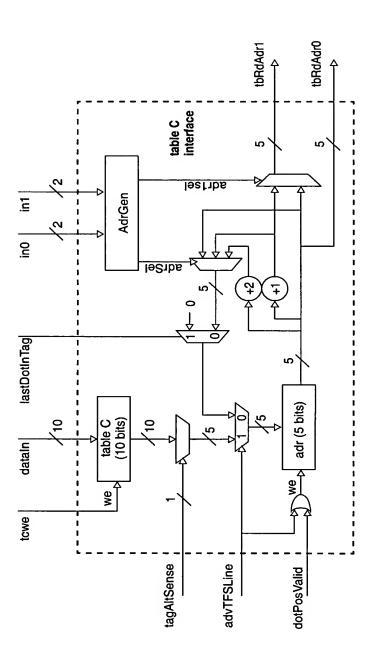


FIG. 226

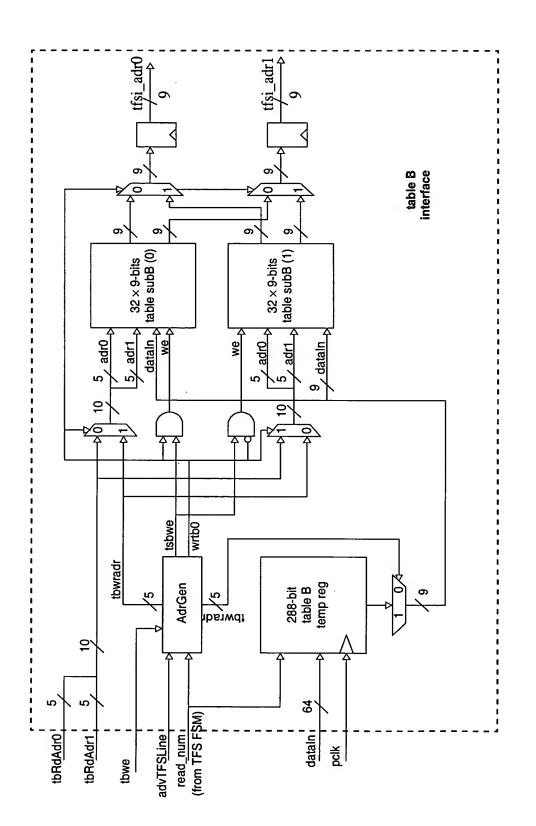
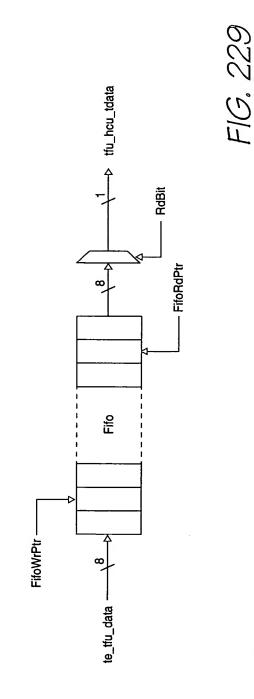


FIG. 227



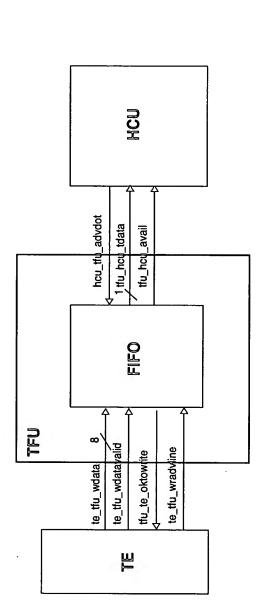


FIG. 228

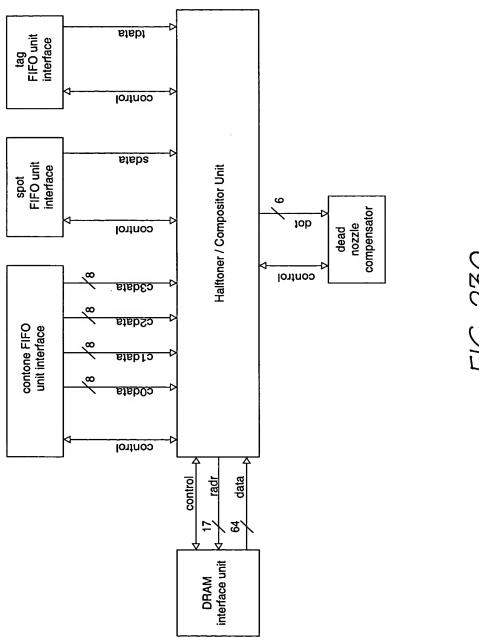


FIG. 230

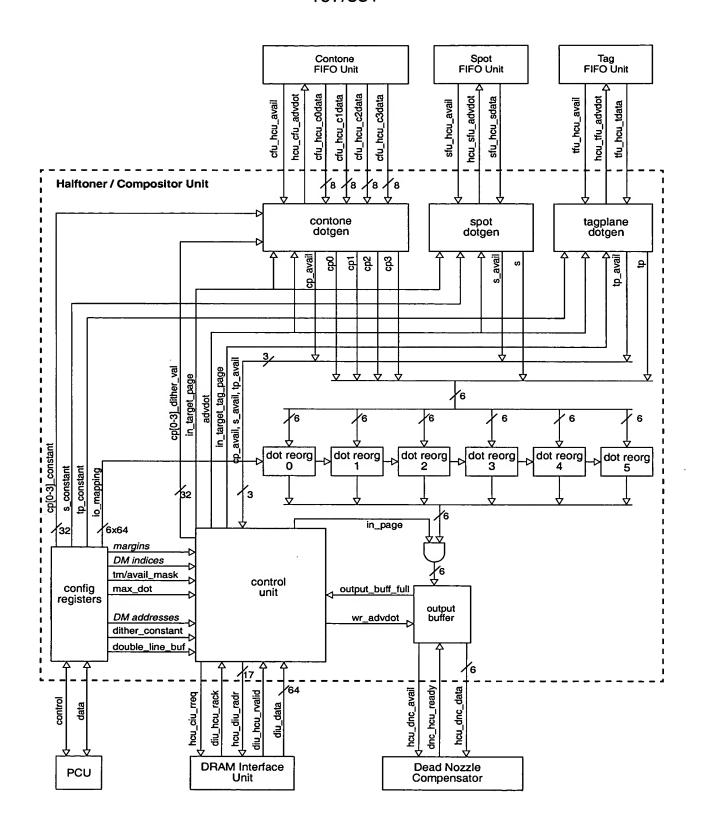
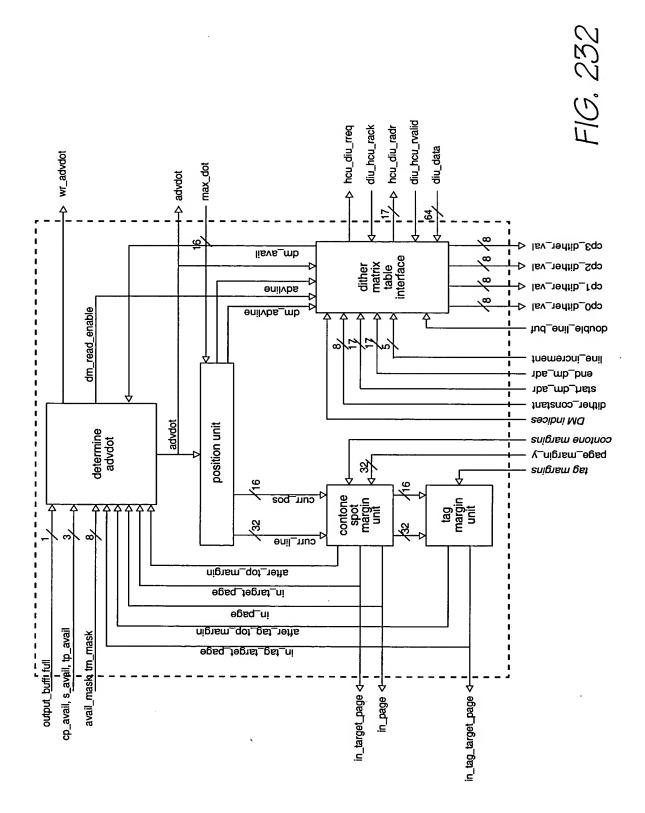
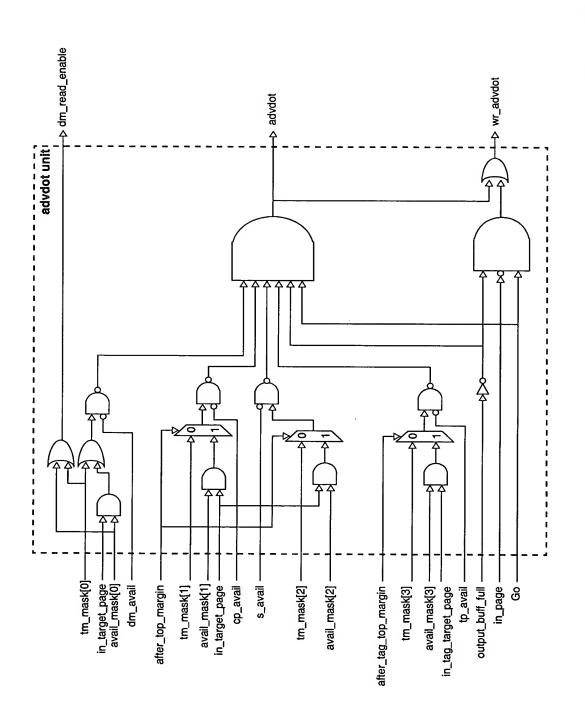


FIG. 231





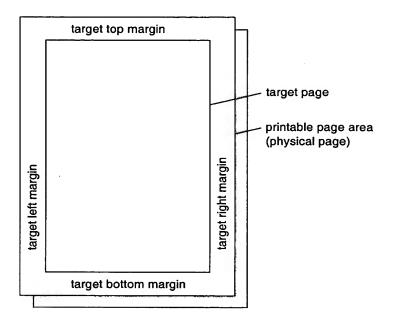


FIG. 234

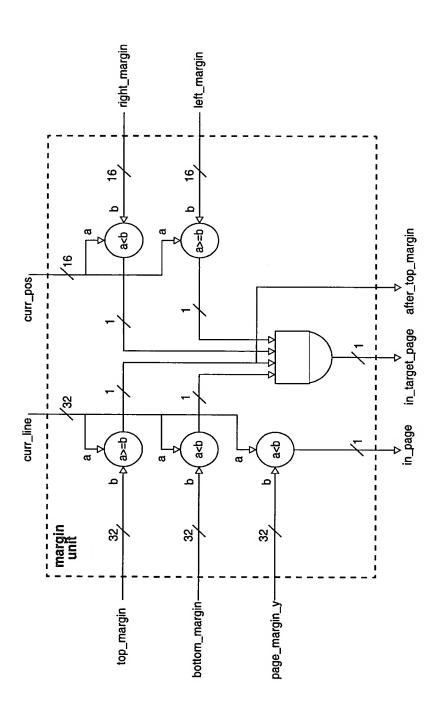
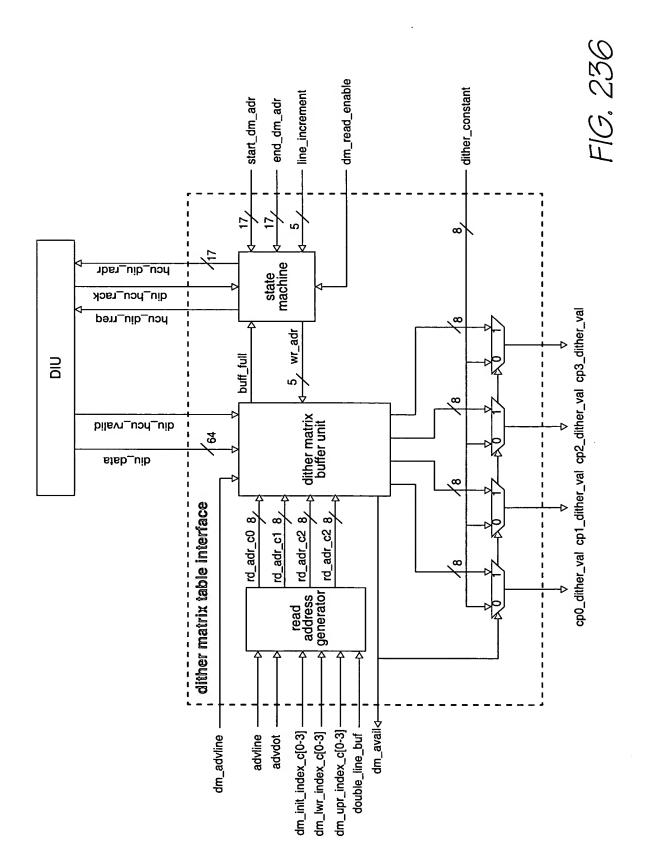


FIG. 235



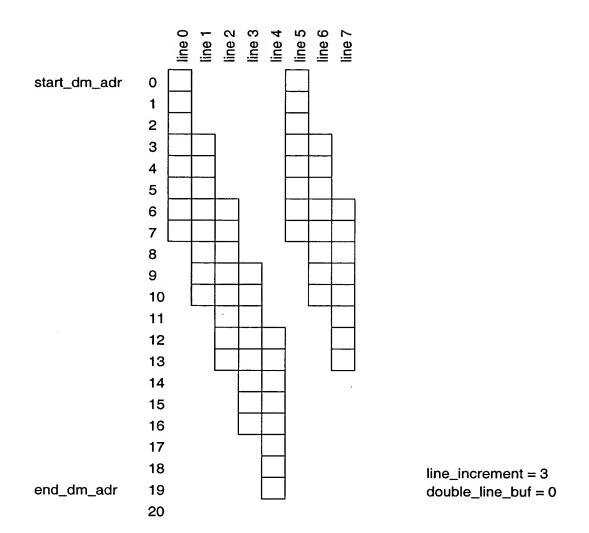


FIG. 237

204/331

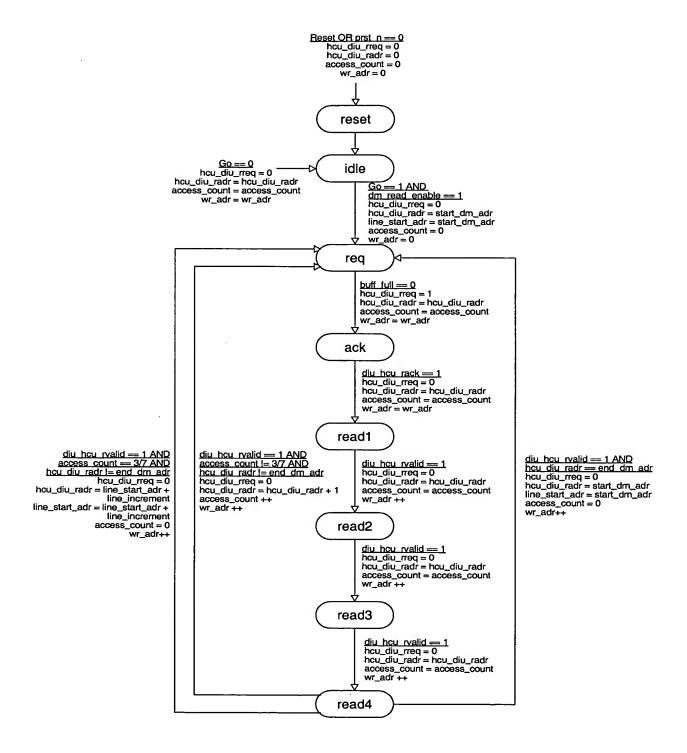


FIG. 238

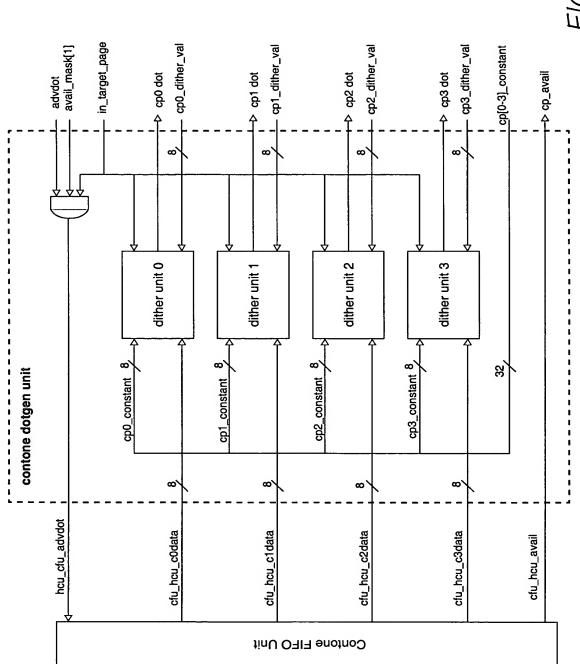


FIG. 239

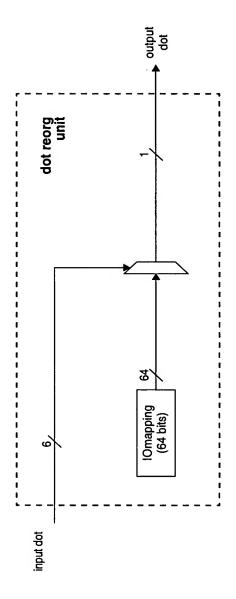
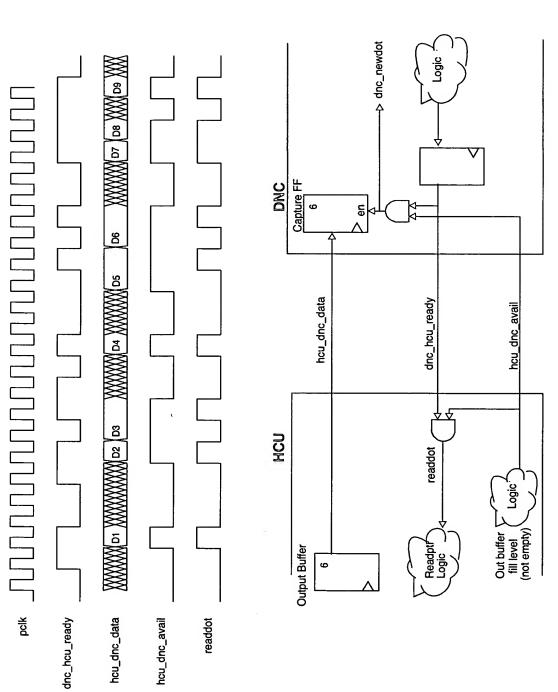


FIG. 240





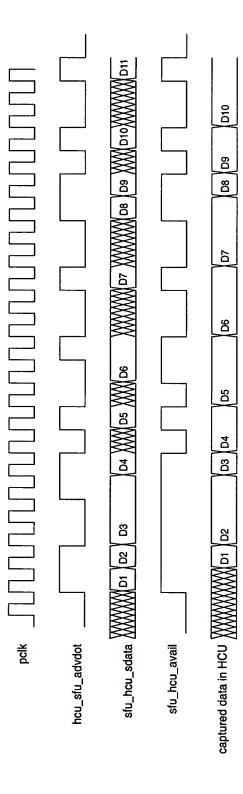


FIG. 242

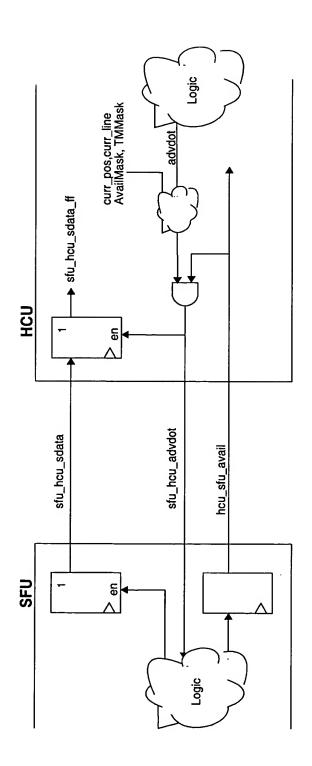


FIG. 243

210/331

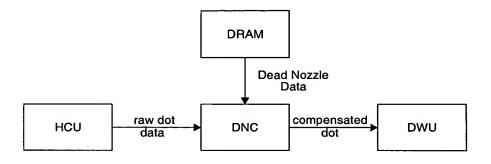


FIG. 244

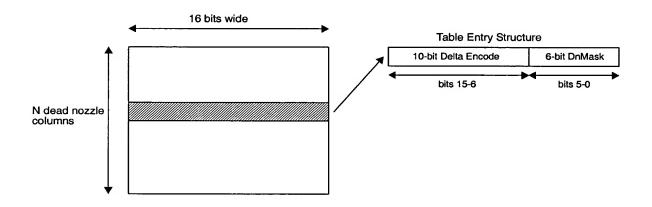


FIG. 245

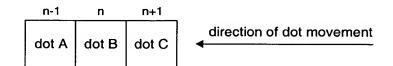


FIG. 246

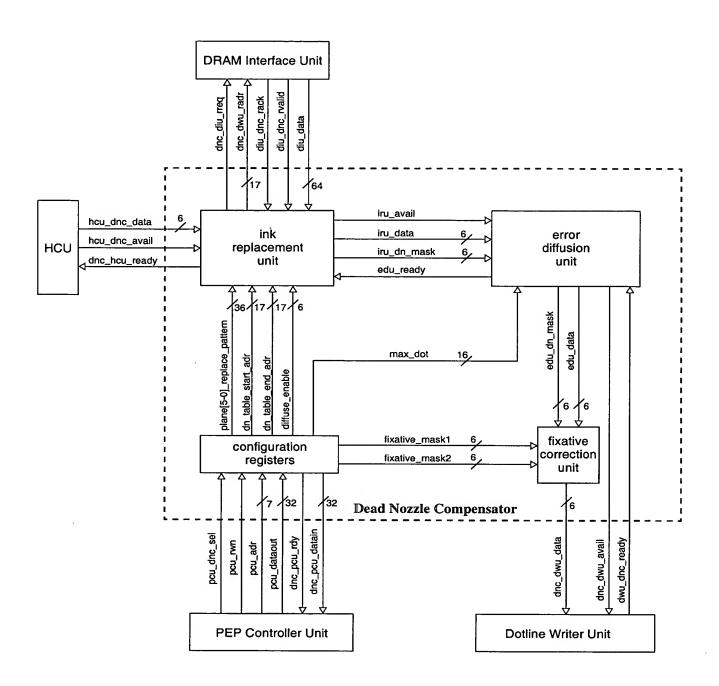


FIG. 247

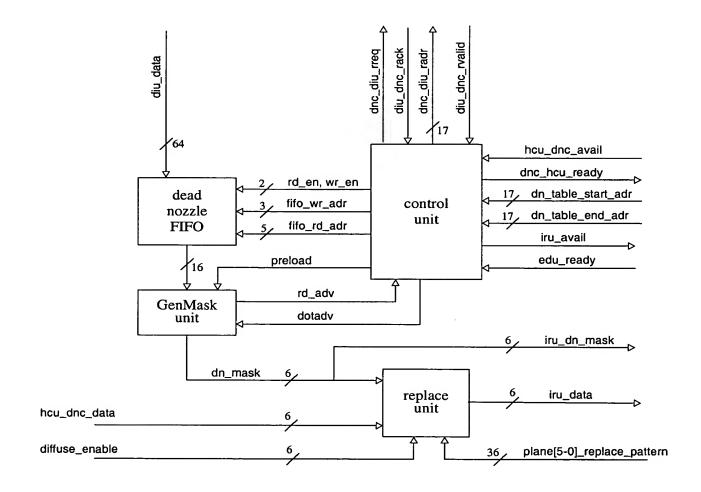


FIG. 248

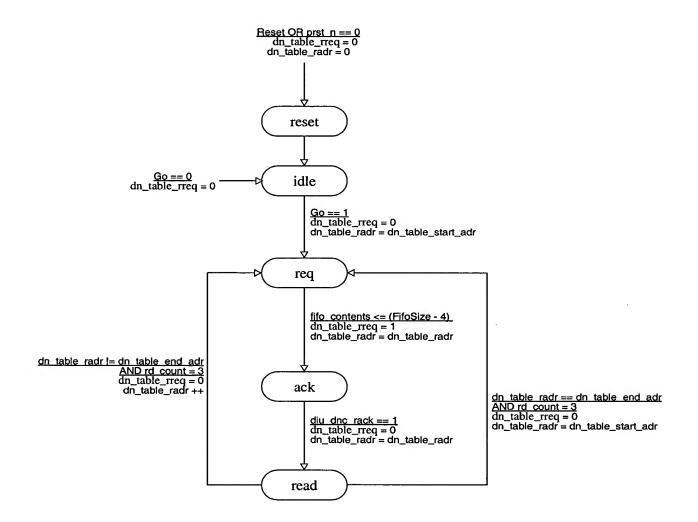


FIG. 249

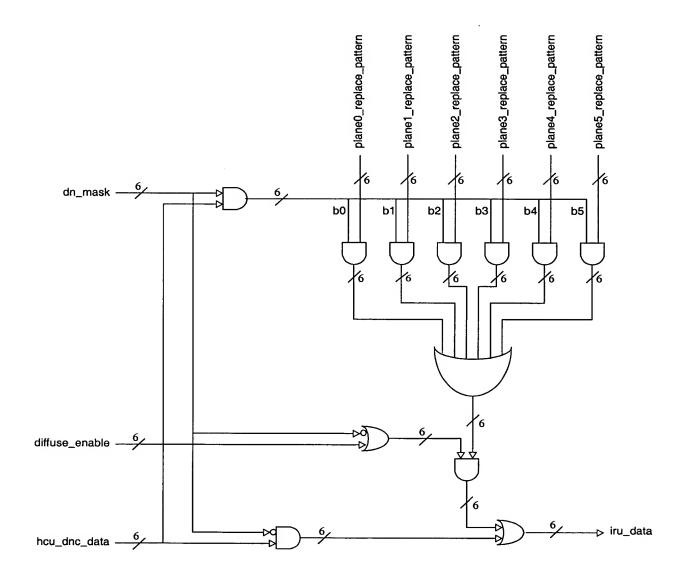


FIG. 250

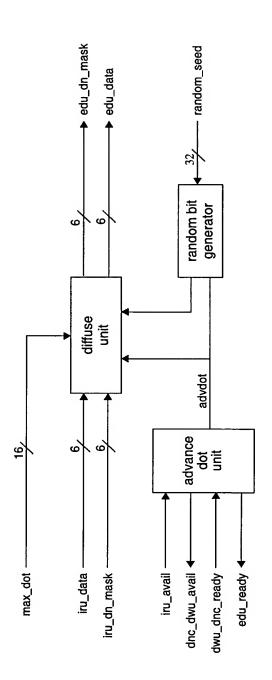
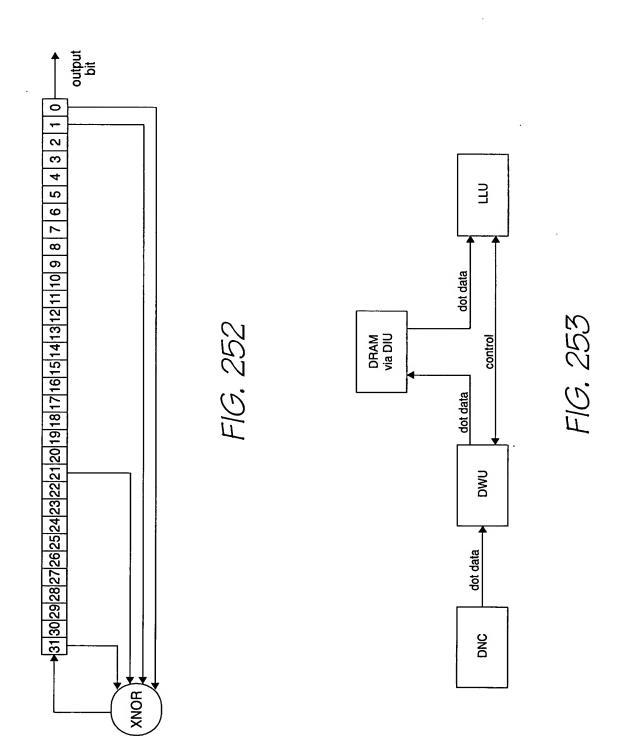
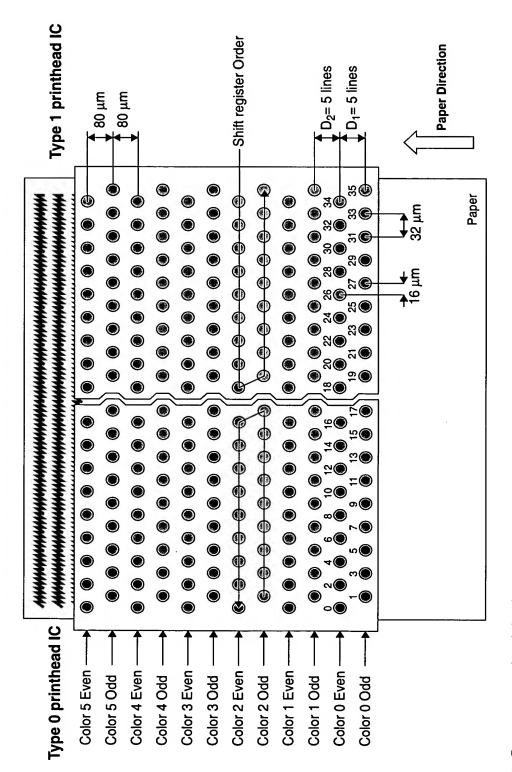


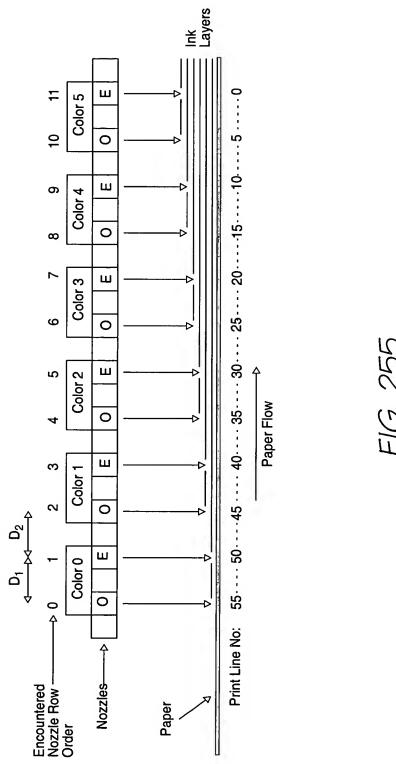
FIG. 251

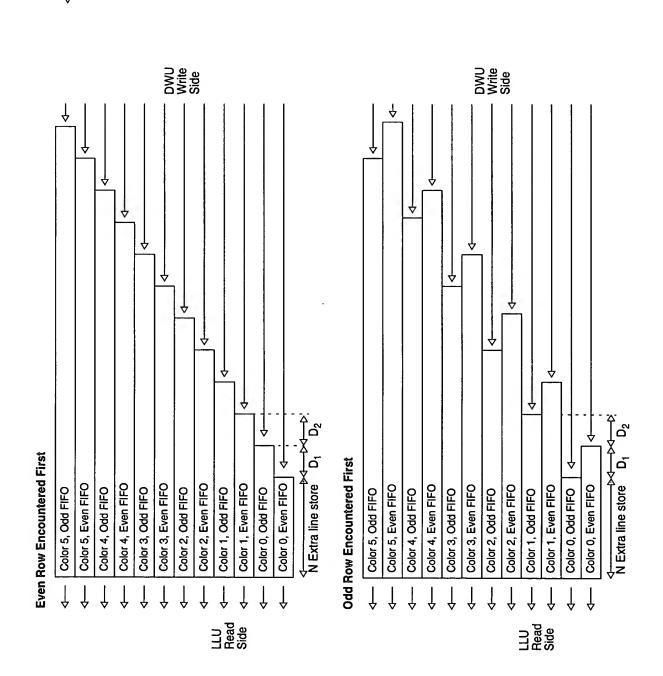


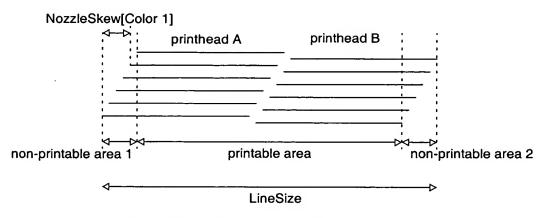


Note: Paper passes under printhead

FIG. 254







non-printable area 1 = inverted non-printable area 2

FIG. 257

printhead A	printhead B

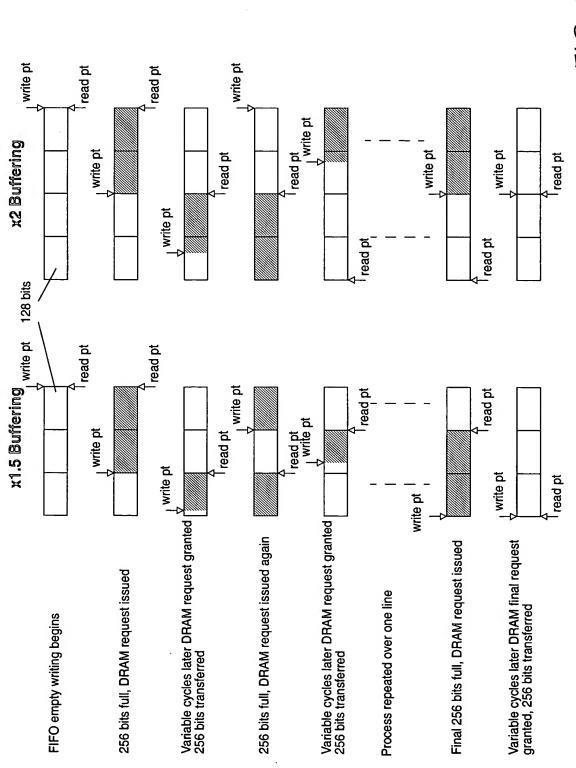


FIG. 259

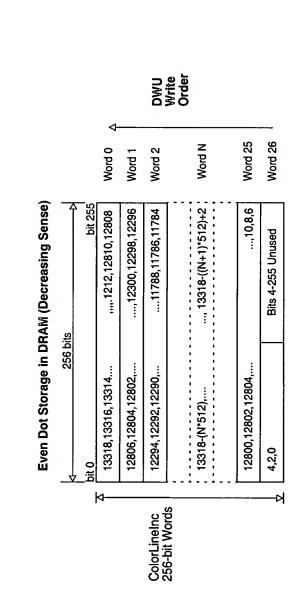


FIG. 260

Word 25 Word 26

...,13306,13308,13310

Bits 4-255 Unused

13312,13314,13316,13318

12800, 12802, 12804,....

Word N

(N+1)*512 - 2

N*512

DWU Write Order

Word 1 Word 2

1024,1026,1028,1030.....

ColorLineInc 256-bit Words

512,514,516,518...

0,2,4,6....

bit 0

Word 0

.....502,504,508,5101016,1018,1020,1022,1528,1530,1532,1534

bit 255

Even Dot Storage in DRAM (Increasing Sense)

256 bits

FIG. 261

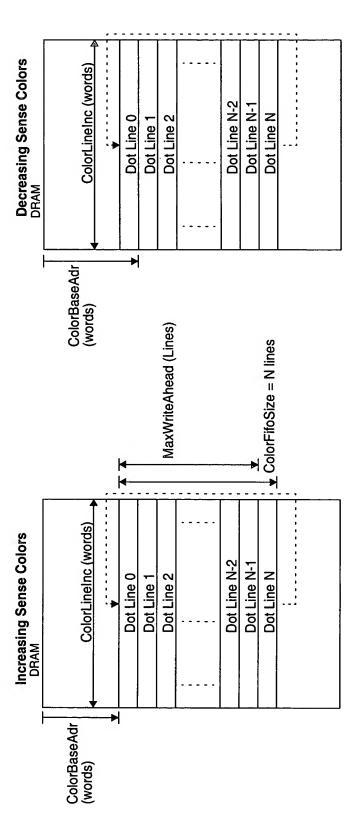


FIG. 262

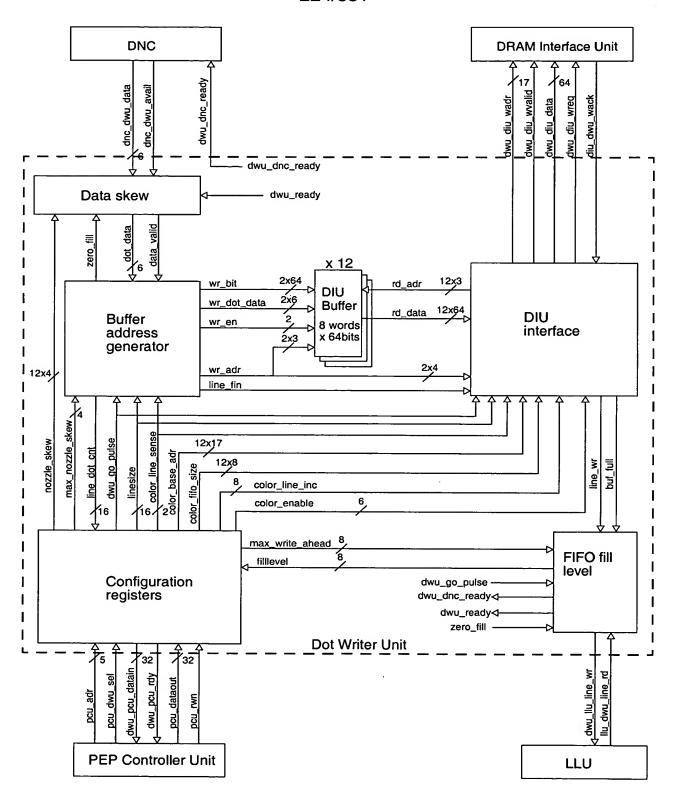
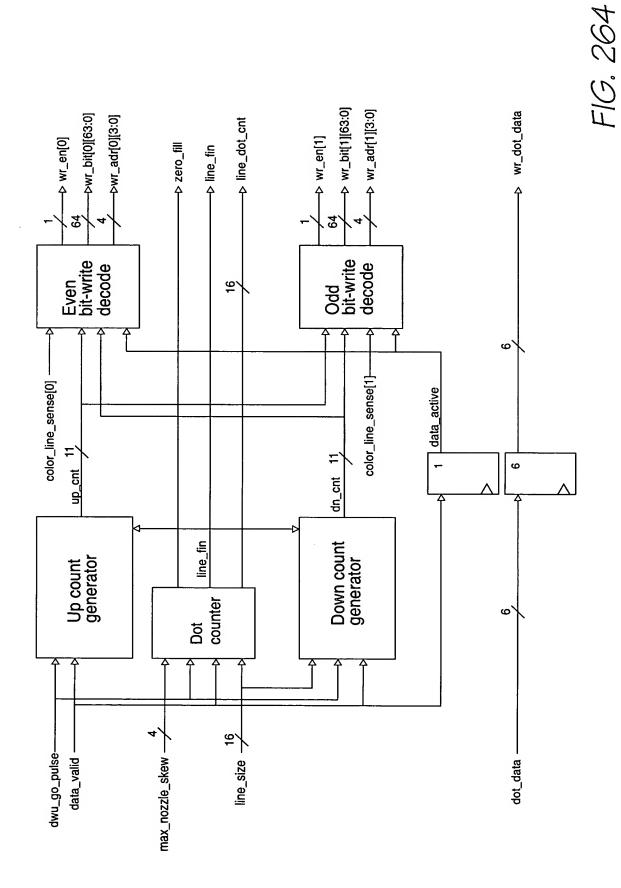
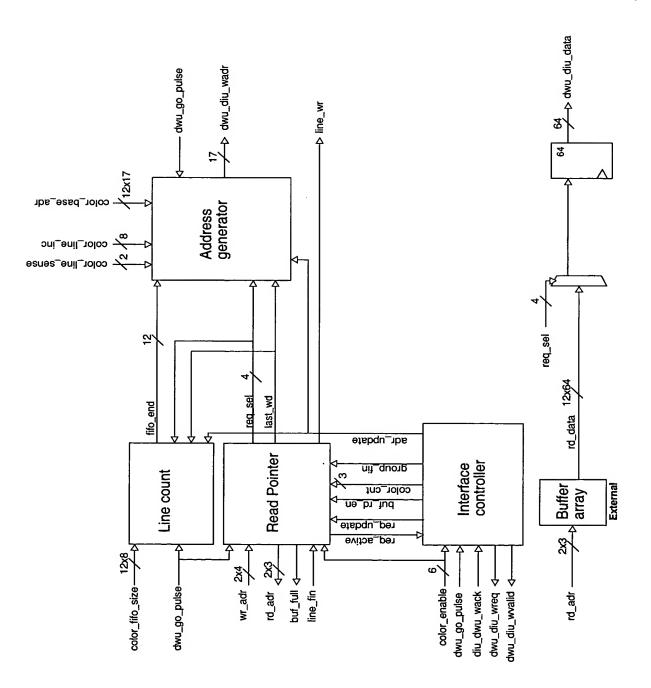
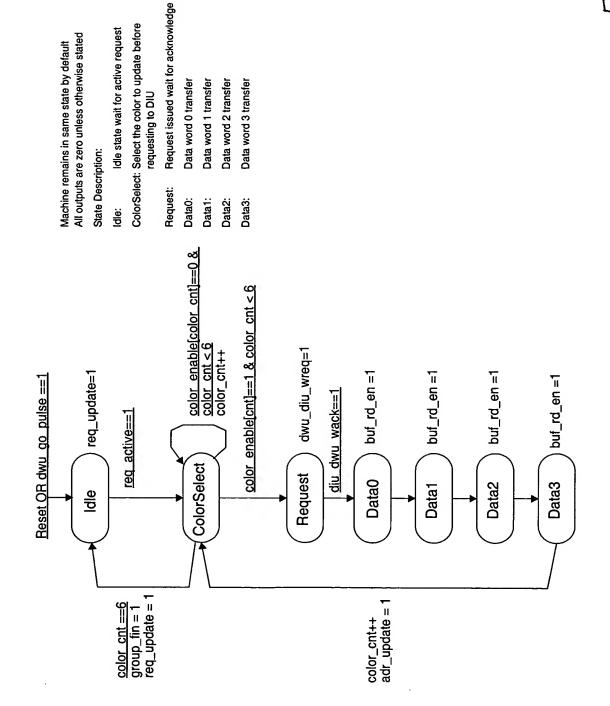


FIG. 263







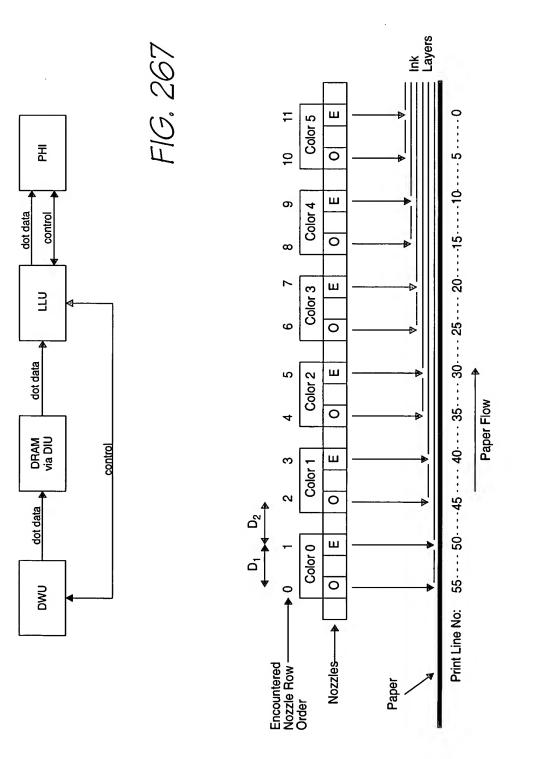
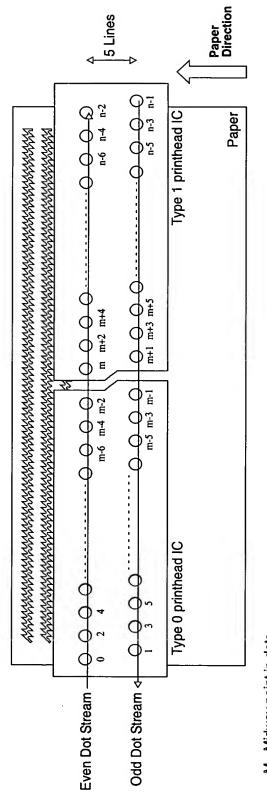


FIG. 268



M - Midway point in dots N - Number of dots in a line

Note: Paper passing under printhead

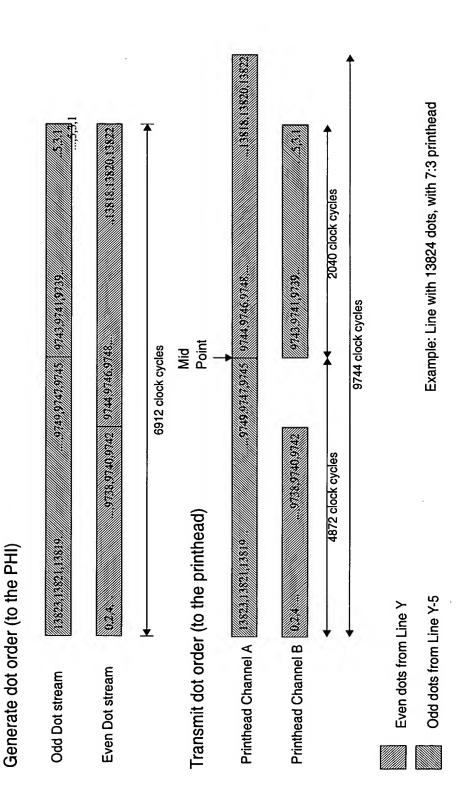


FIG. 270

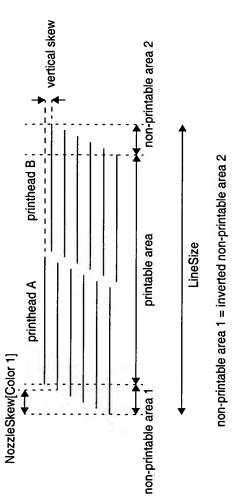


FIG. 271

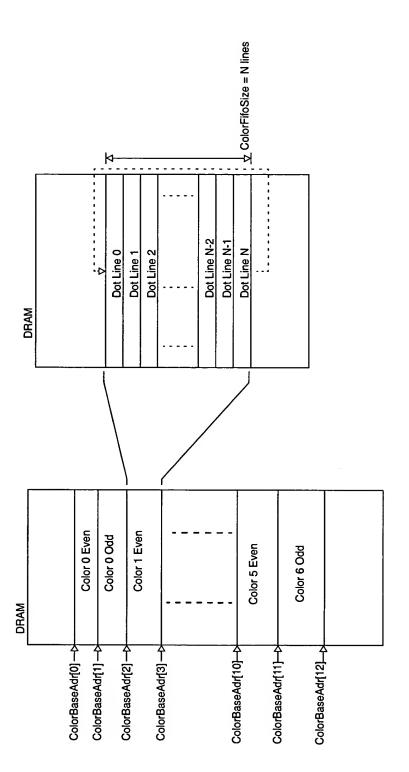
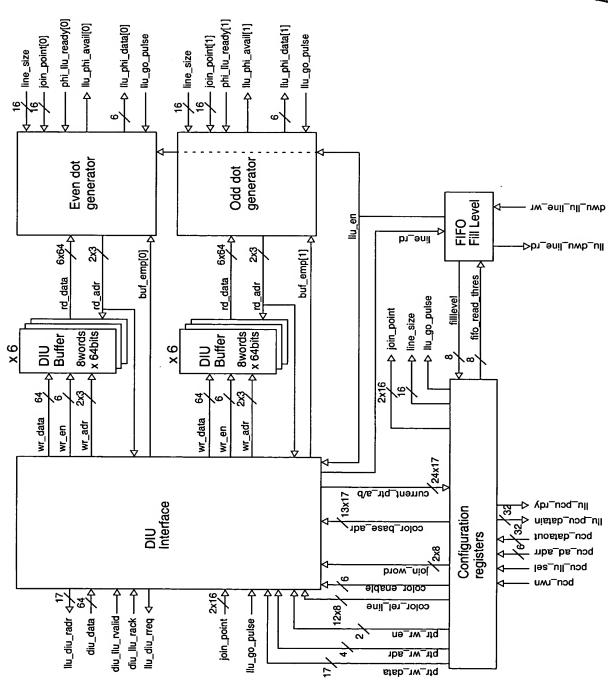


FIG. 272



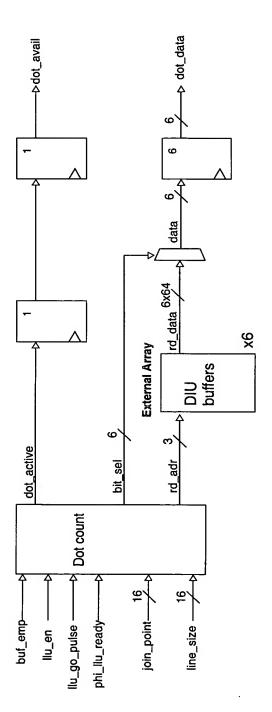
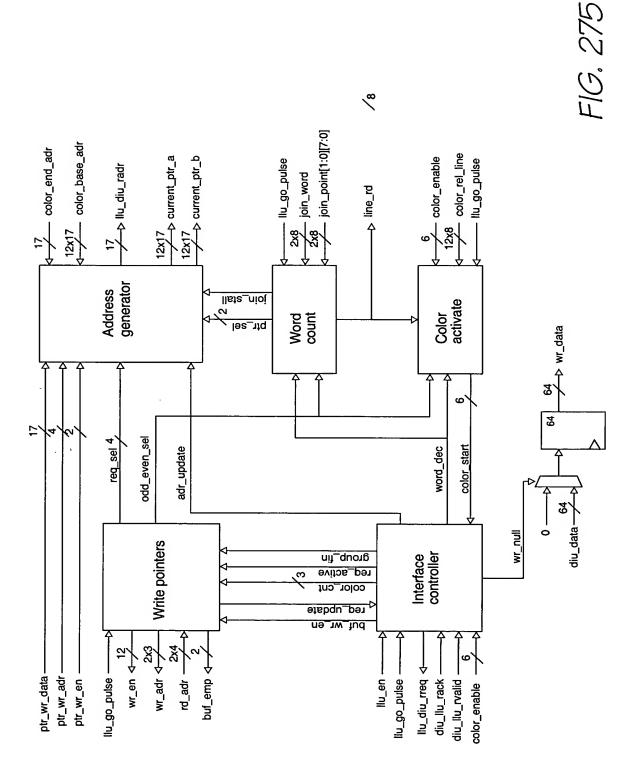
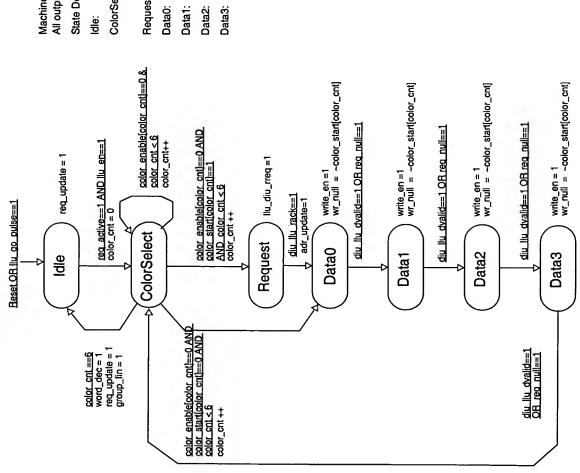


FIG. 274







Machine remains in same state by default All outputs are zero unless otherwise stated

State Description:

Idle: Idle state wait for active request

ColorSelect: Select the color to update before requesting to DIU

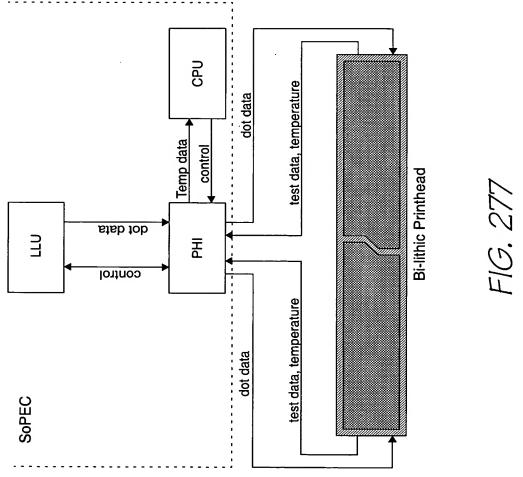
Request: Request issued wait for acknowledge

sta0: Data word 0 transfer

Data1: Data word 1 transfer

Data2: Data word 2 transfer

Data3: Data word 3 transfer



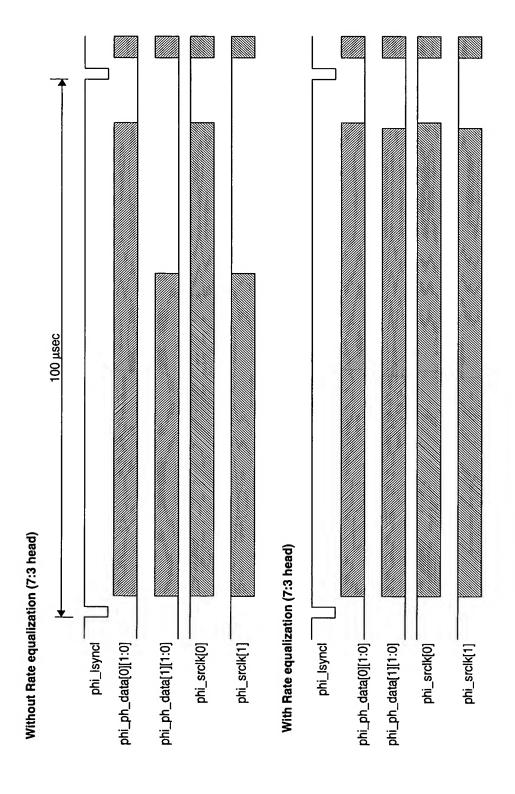
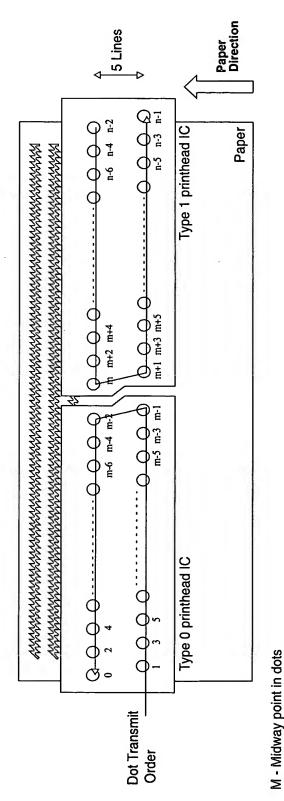
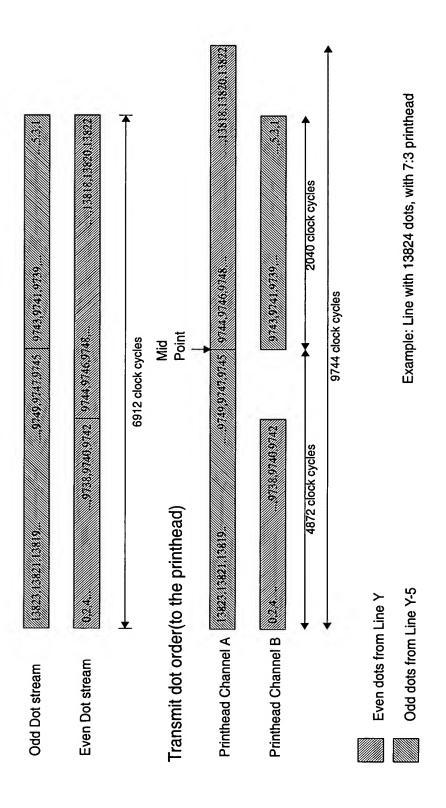


FIG. 279



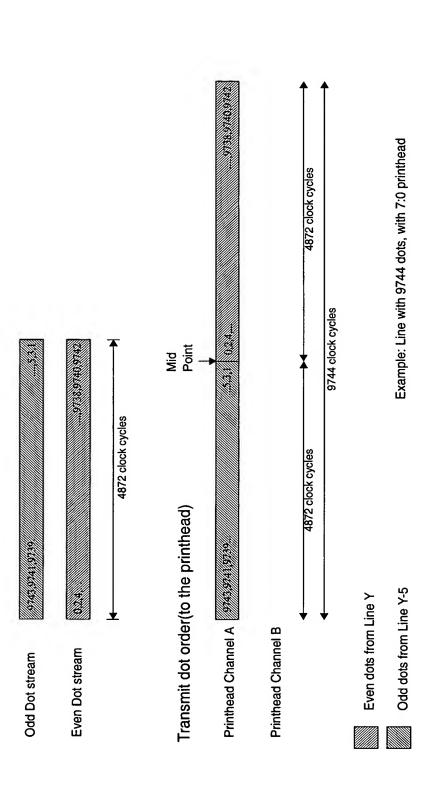
N - Number of dots in a line

Note: Paper passing under printhead



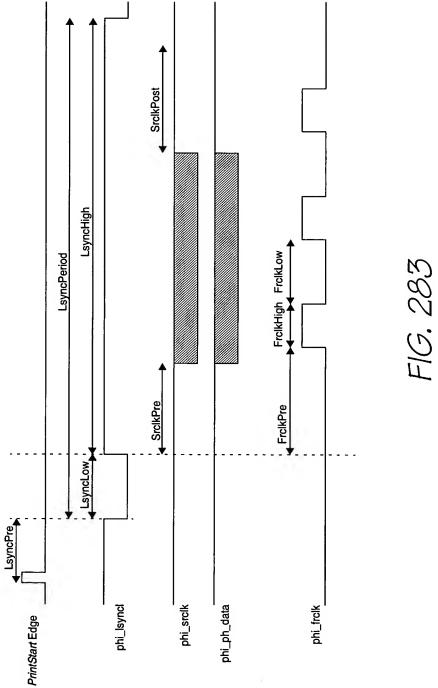
Generate dot order (from the LLU)

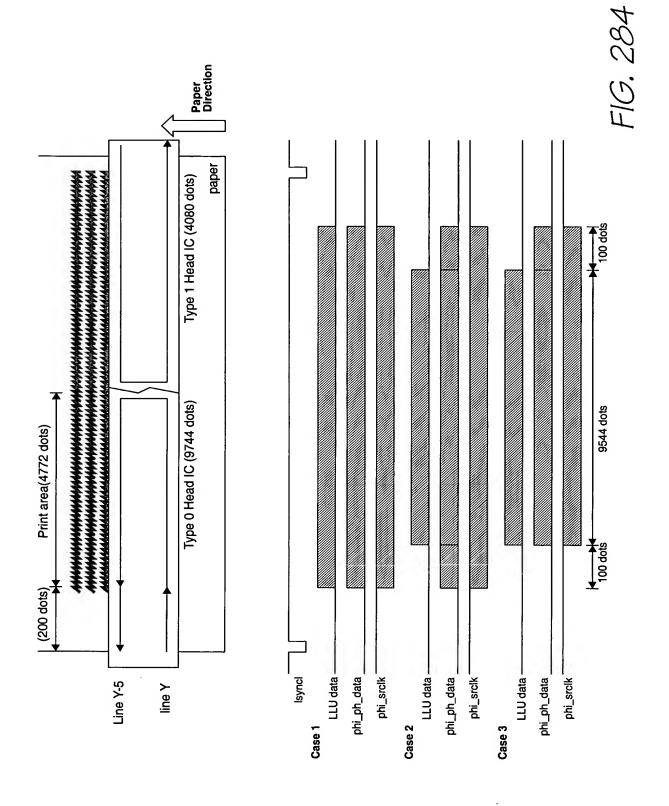
FIG. 281



Generate dot order (from the LLU)

FIG. 282





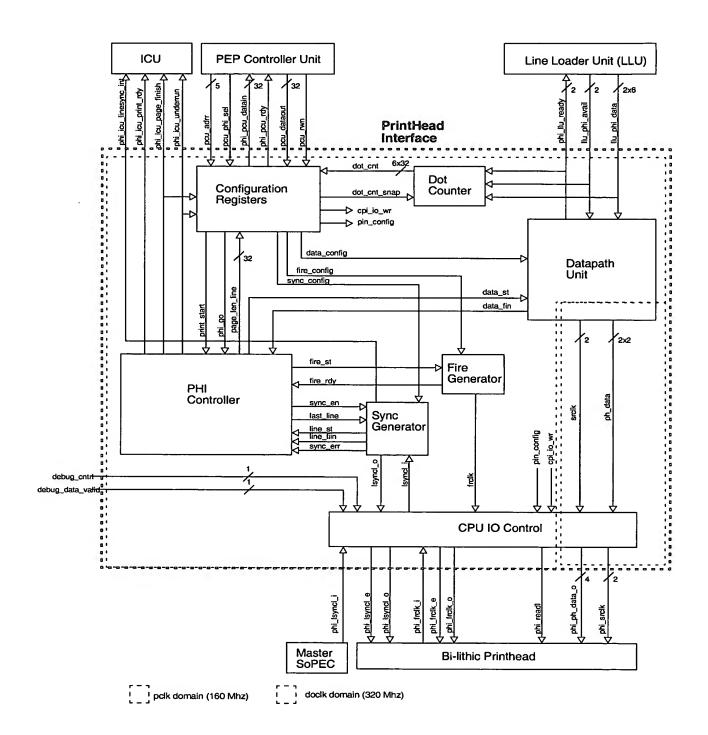
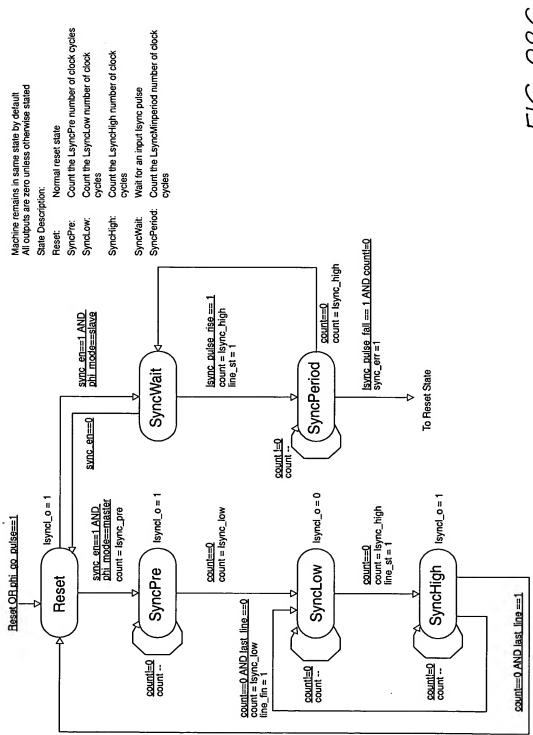
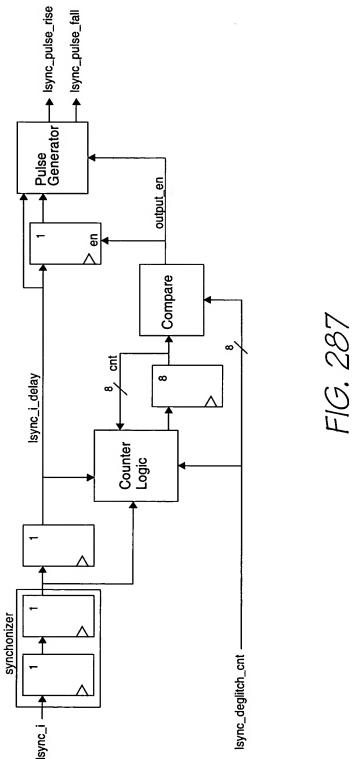


FIG. 285











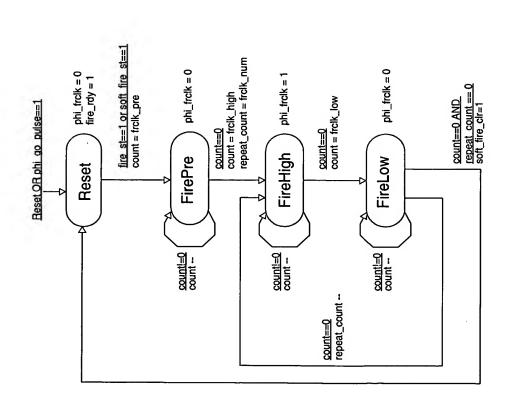
State Description:

Normal reset state Reset:

Count the FrclkPre number of clock cycles, repeat count set to FrclkNum FirePre:

Count the FrclkHigh number of clock cycles FireHigh:

Count the FrclkLow number of clock cycles FireLow:



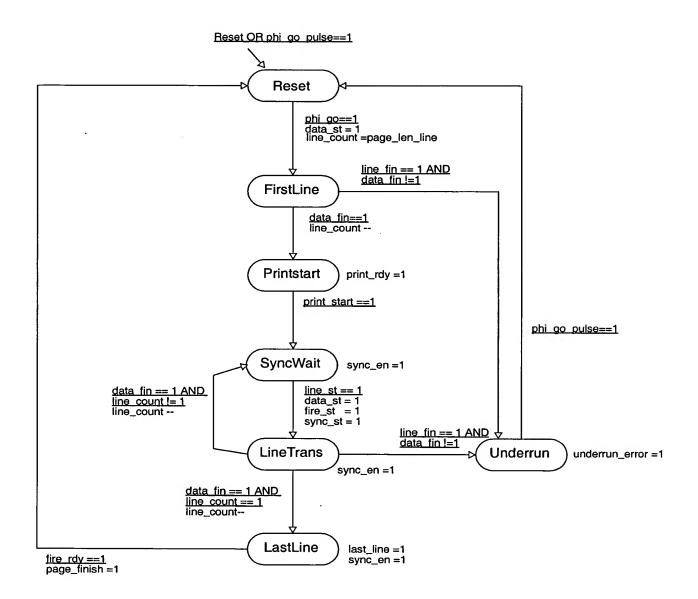


FIG. 289

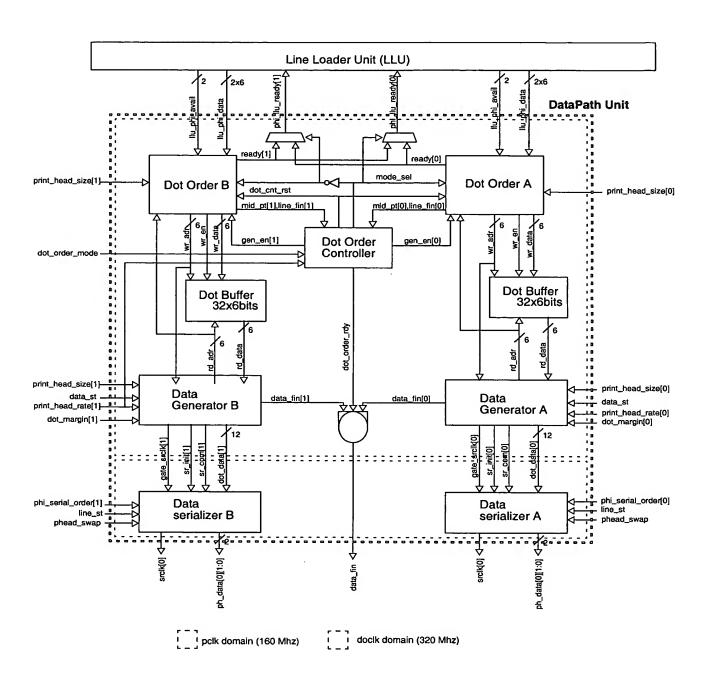
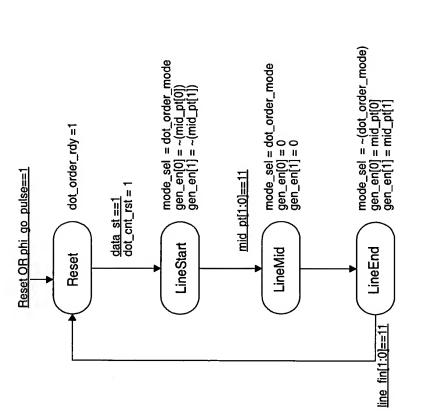


FIG. 290



Line end processing wait for both line_fin to be active

LineMid: LineEnd:

Start processing first part of the line, wait for both mid_pt to be active
Switch over wait state allow pipeline to clear

Linestart:

Normal reset state

Reset:

Machine remains in same state by default All outputs are zero unless otherwise stated

State Description:



All outputs are zero unless otherwise stated Machine remains in same state by default

State Description:

Reset:

Normal reset state

Count the SrclkPre number of clock

cycles SrclkPre:

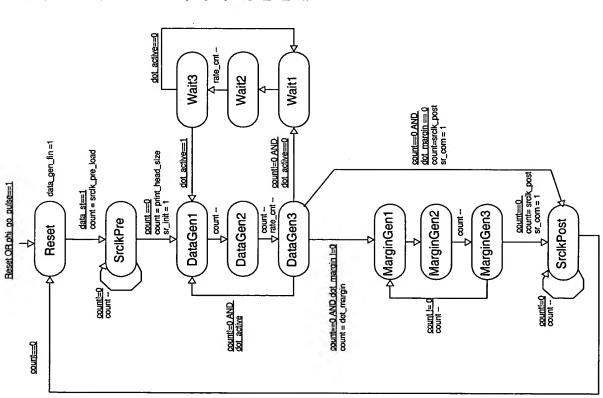
DataGen1: Read Line Dot data from buffer DataGen2: Read Line Dot data from buffer

Data rate control wait state DataGen3: Data gen wait state

Data rate control wait state Wait1: Wait2: Data rate control wait state Wait3:

MarginGen1:Generate DotMargin number of dots MarginGen2:Generate DotMargin number of dots

SrclkPost: Wait for SrclkPost number of cycles MarginGen3:Generate margin wait state



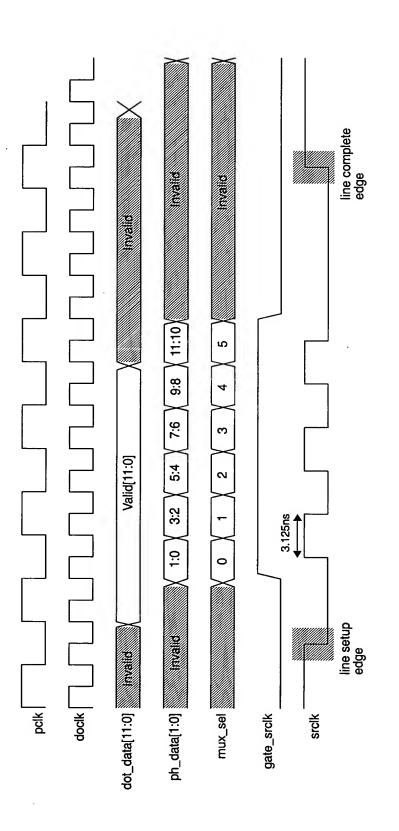


FIG. 293

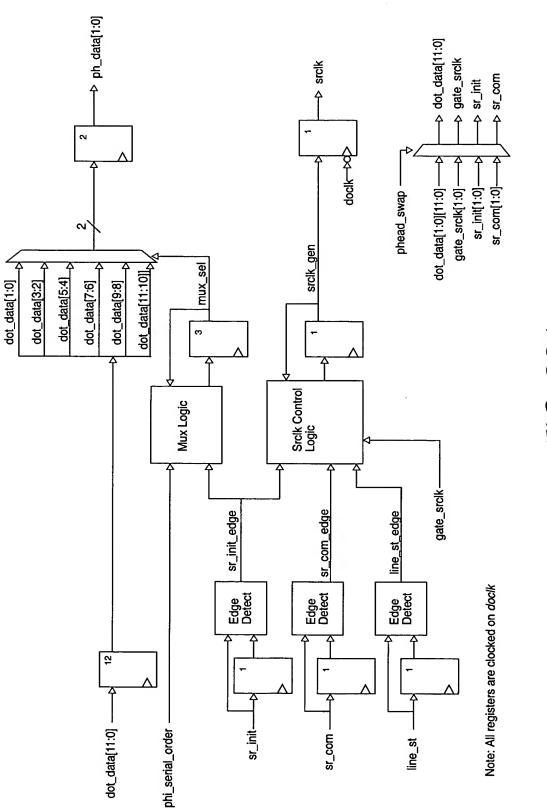


FIG. 294

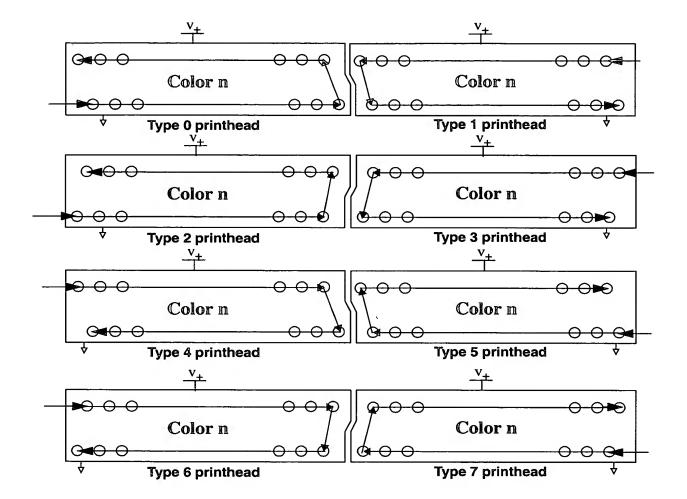


FIG. 295

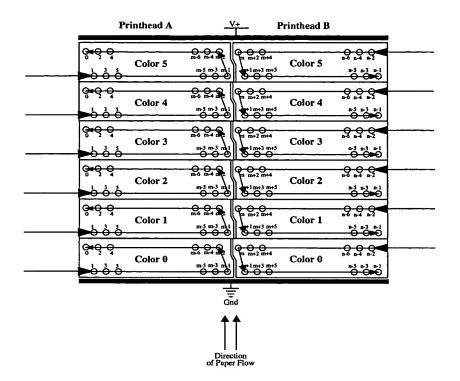


FIG. 296

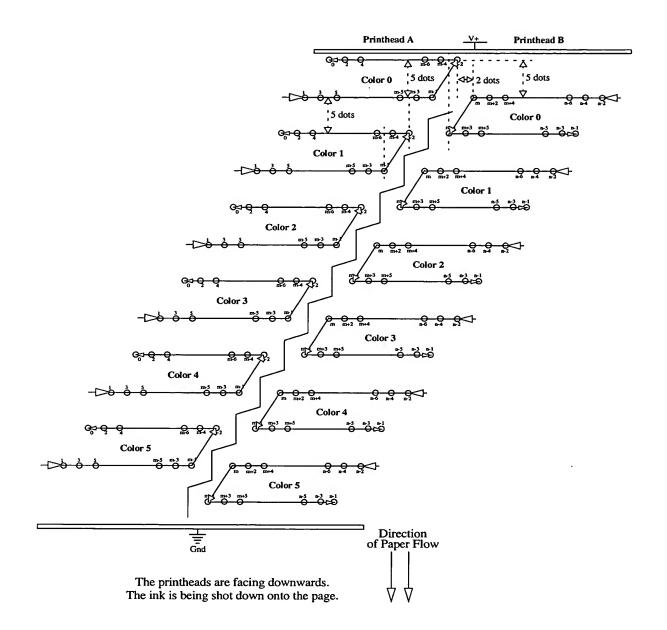


FIG. 297

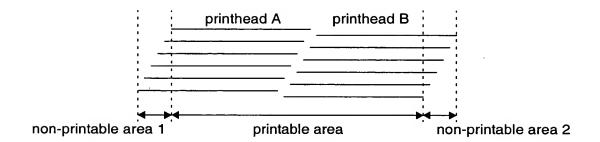


FIG. 298

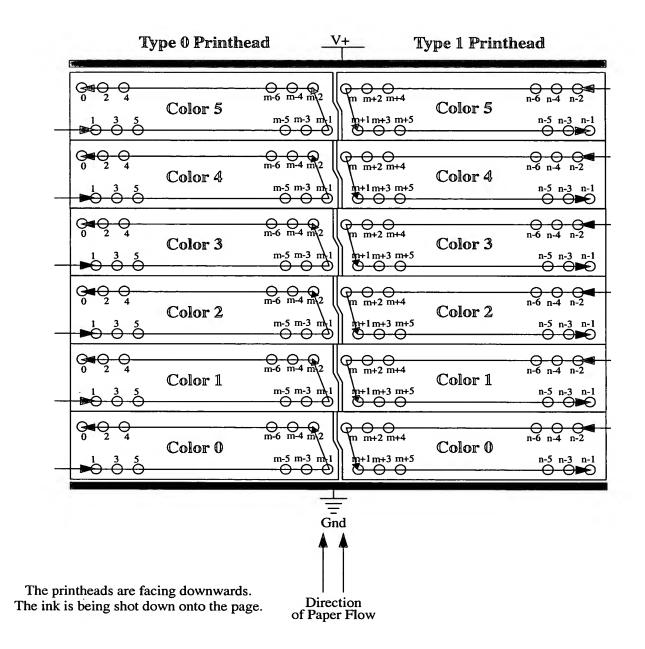


FIG. 299

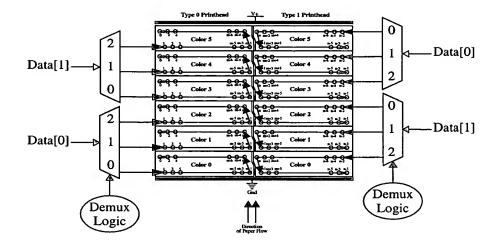


FIG. 300

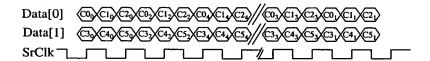


FIG. 301

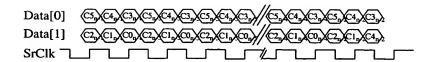


FIG. 302

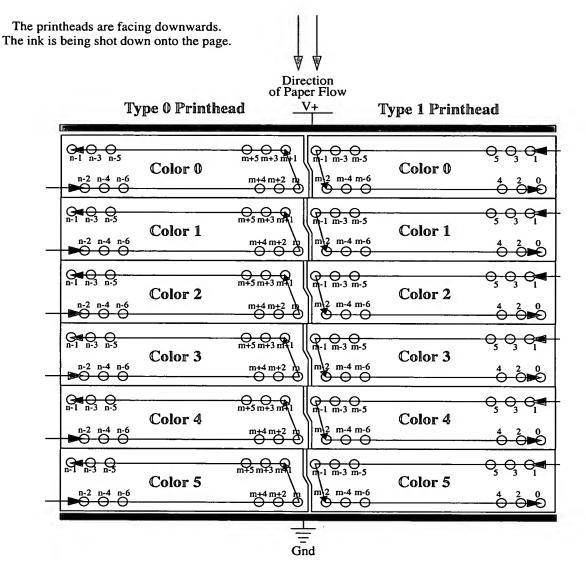


FIG. 303

261/331

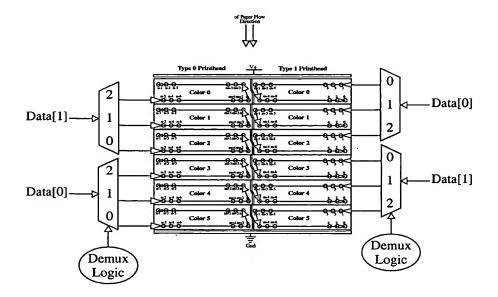


FIG. 304

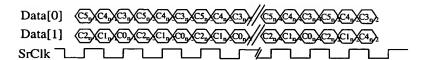


FIG. 305

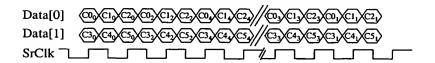


FIG. 306

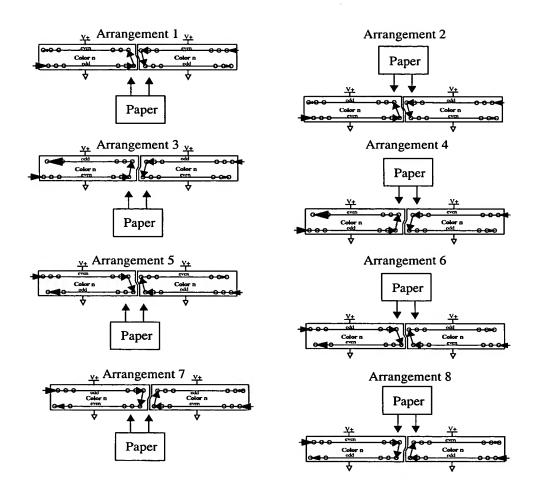


FIG. 307

FIG. 308

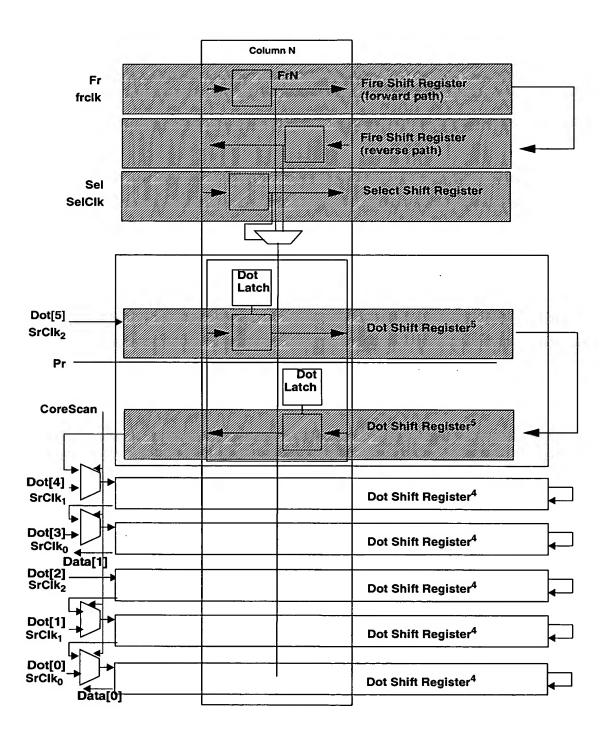


FIG. 309

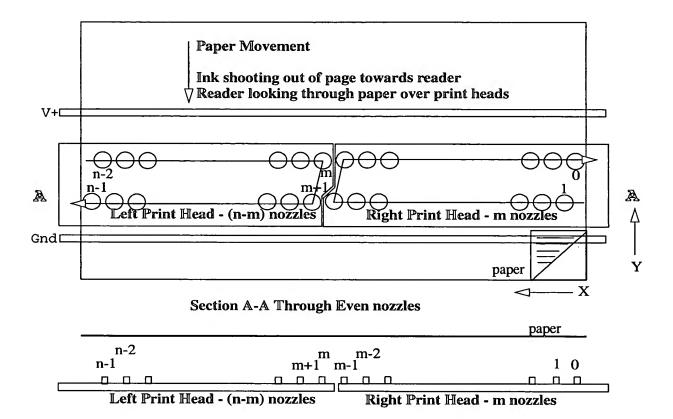


FIG. 310

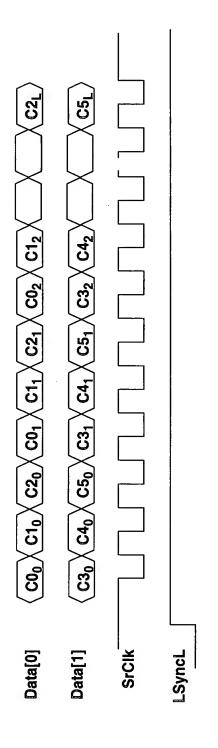


FIG. 311

FIG. 312

J─_fire shift] 👞 register	select shift reg
100000000001000000000010000000001000000	0000000000100000000010000000001000000	000000000001111111111110000000

FIG. 313

1000000000100000000010000000001000000	00000000001000000000100000000	111111111100000000001111111111000000	Right Print Head Fire/Select SR
0	OT A	11	귚
	₽		
占	┻,		
읽	2	11	æ
000	001	111	ct SR
0001000000000	0010000000001	00011111111111	lead Fire/Select SR
.00010000000001000000000	.00100000000010000000001	.1110000000000011111111111	Left Print Head Fire/Select SR
00010000000001000000001000000000	001000000000100000000010000000	000000000011111111110000000000111111	Left Print Head Fire/Select SR

FIG. 314

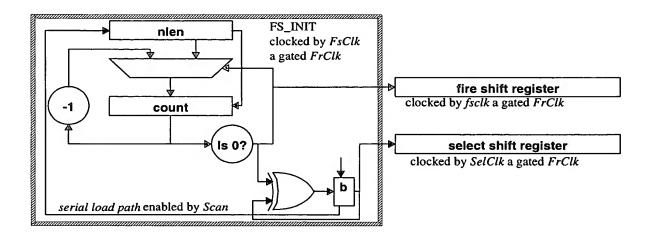


FIG. 315

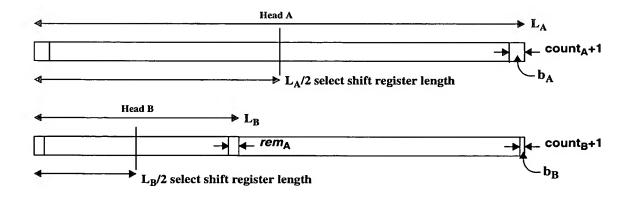


FIG. 316

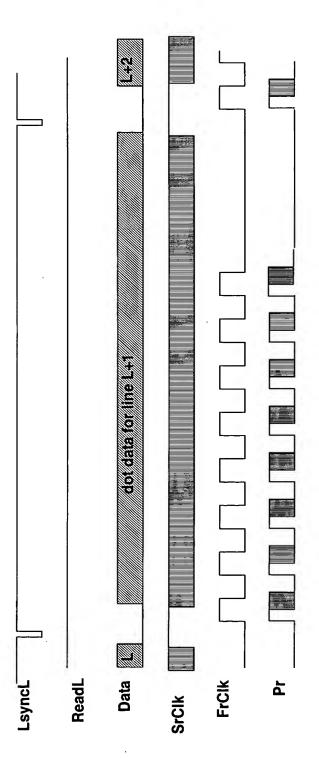


FIG. 317

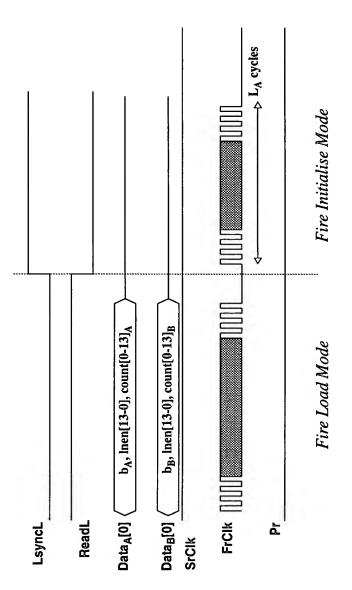
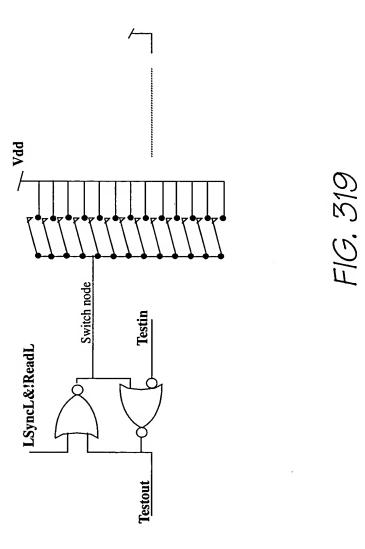


FIG. 318



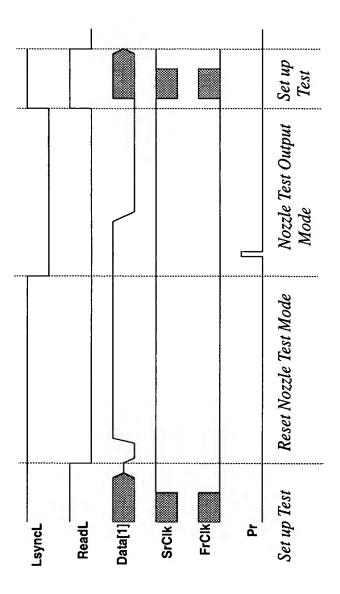


FIG. 320

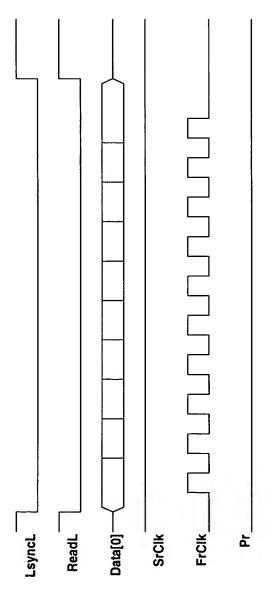


FIG. 321

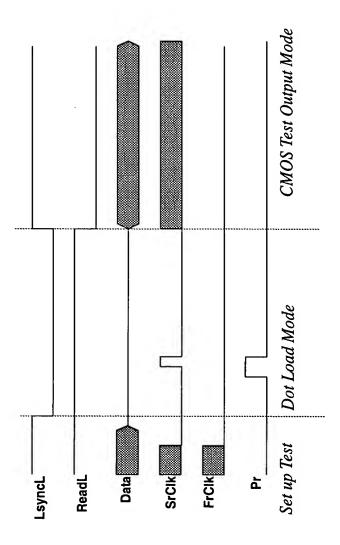


FIG. 322

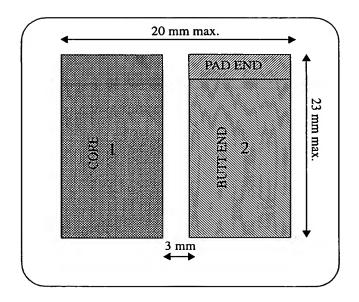


FIG. 323

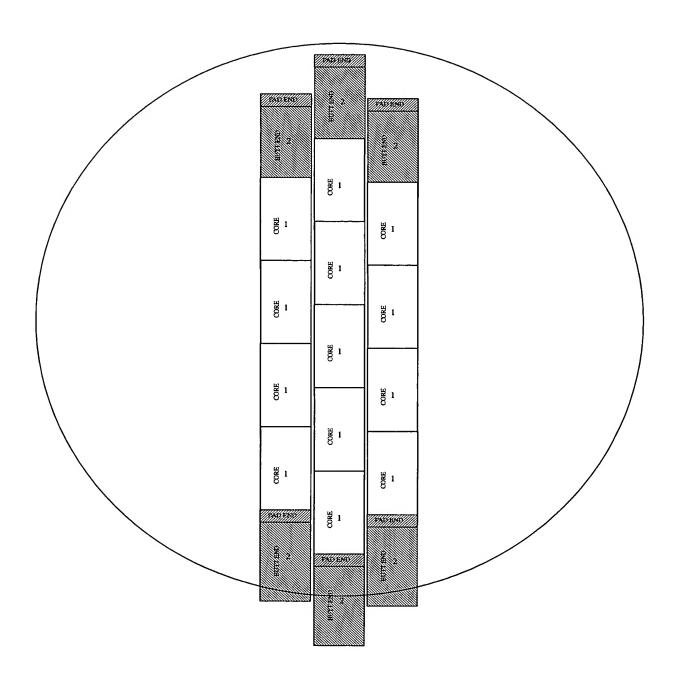
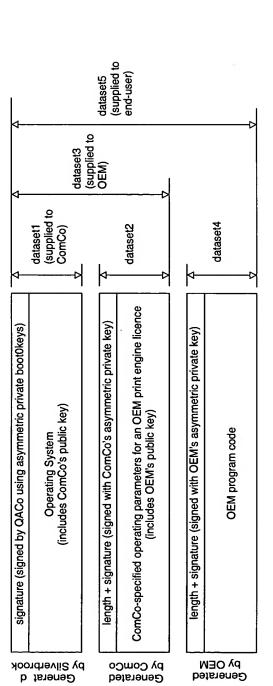
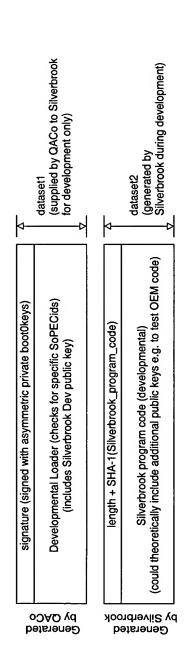


FIG. 324

FIG. 325





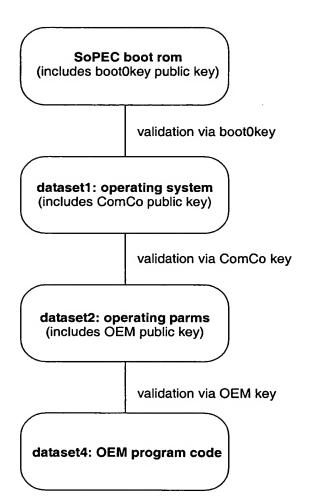


FIG. 326

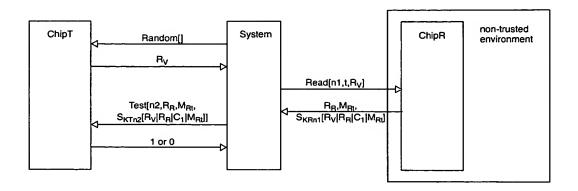


FIG. 328

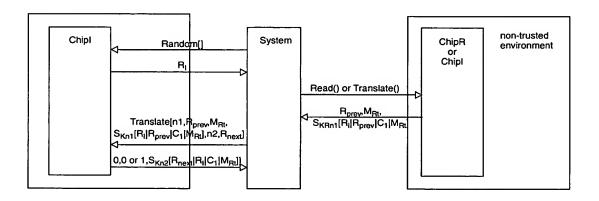


FIG. 329

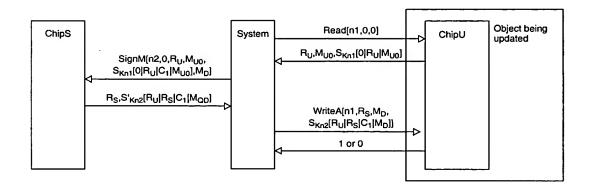


FIG. 330

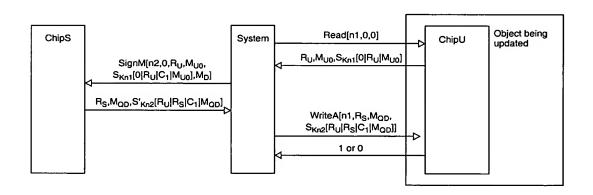


FIG. 331

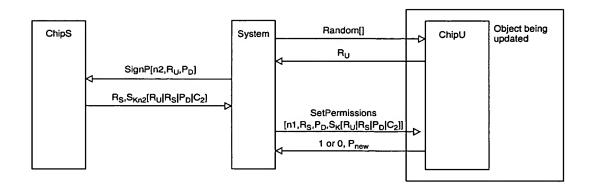


FIG. 332

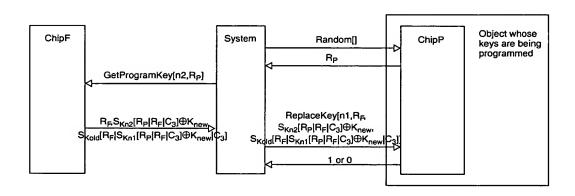


FIG. 333

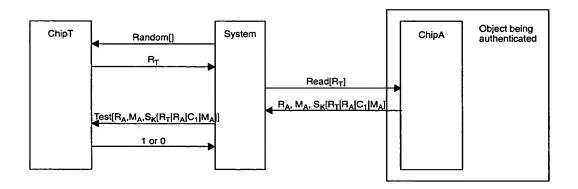


FIG. 334

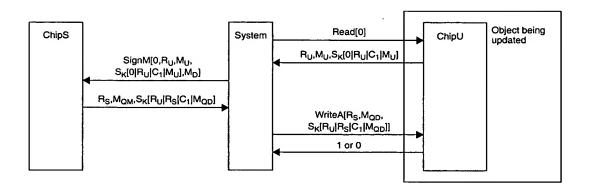


FIG. 335

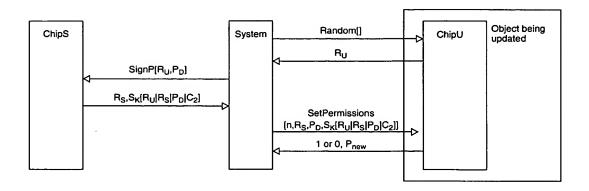


FIG. 336

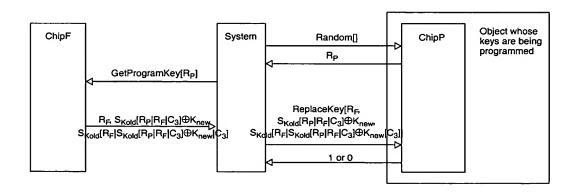


FIG. 337

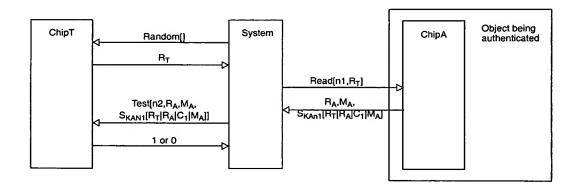


FIG. 338

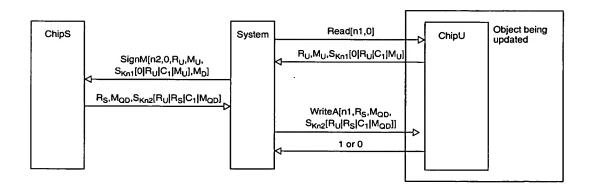


FIG. 339

286/331

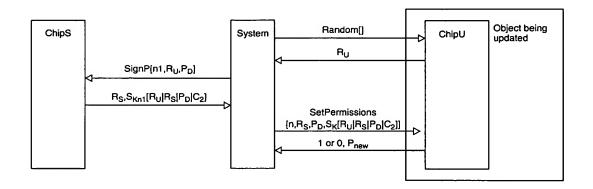


FIG. 340

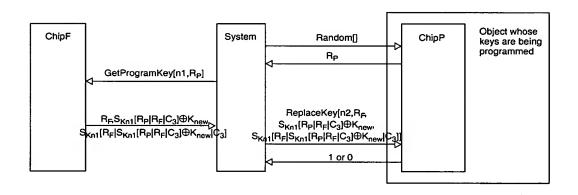


FIG. 341

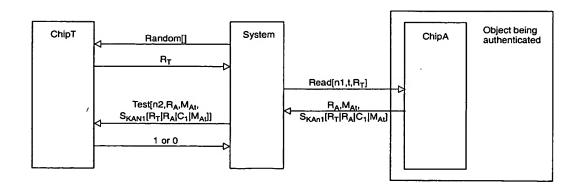


FIG. 342

287/331

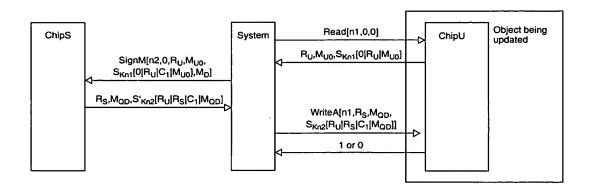


FIG. 343

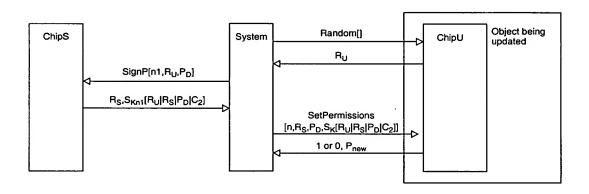


FIG. 344

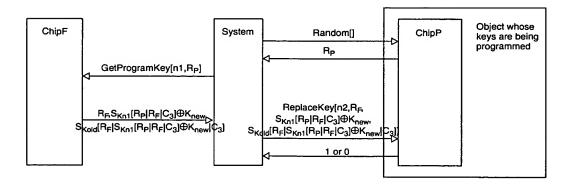


FIG. 345

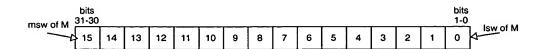


FIG. 346

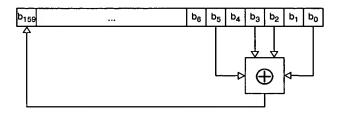


FIG. 347

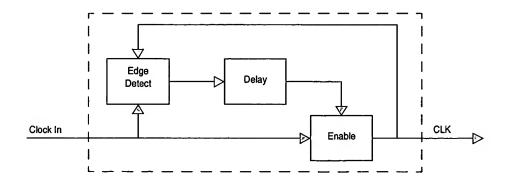


FIG. 348

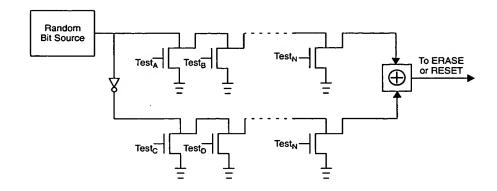


FIG. 349

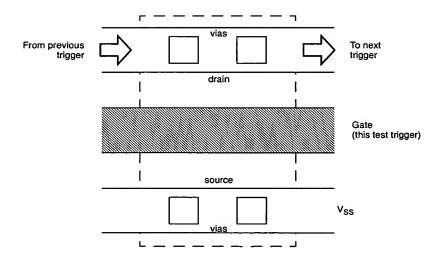


FIG. 350

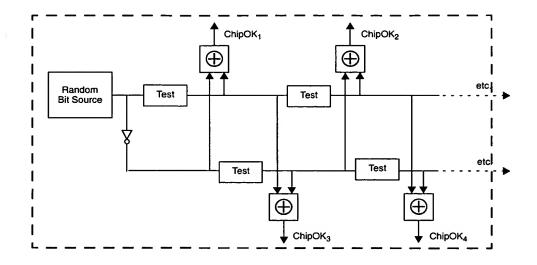


FIG. 351

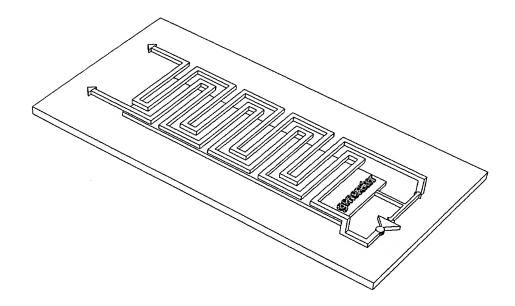


FIG. 352

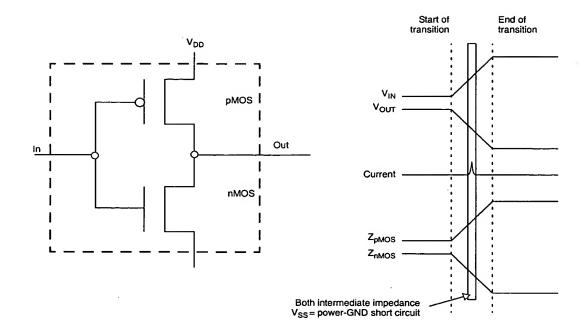


FIG. 353

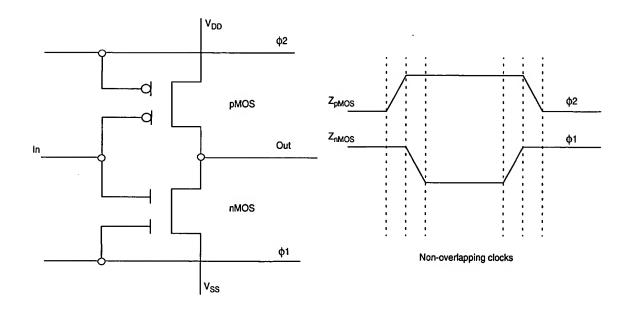


FIG. 354

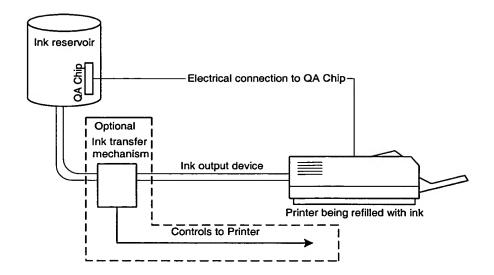


FIG. 355

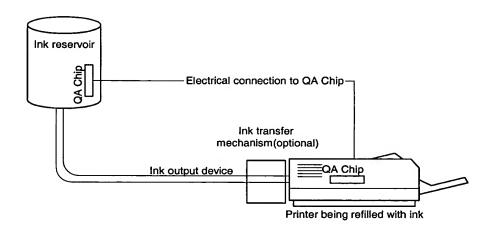


FIG. 356

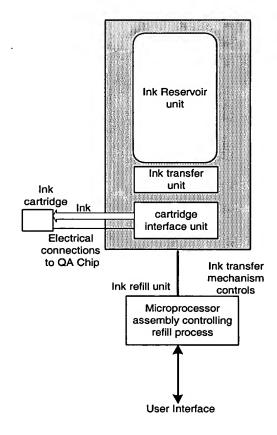


FIG. 357

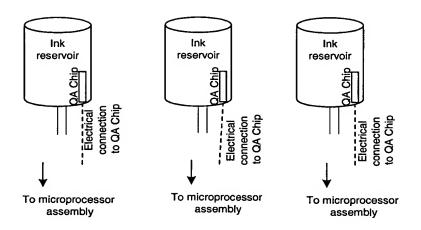


FIG. 358

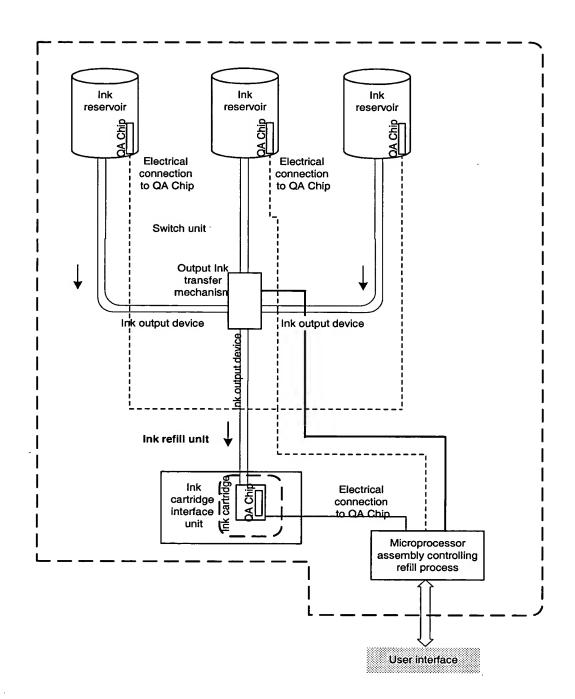


FIG. 359

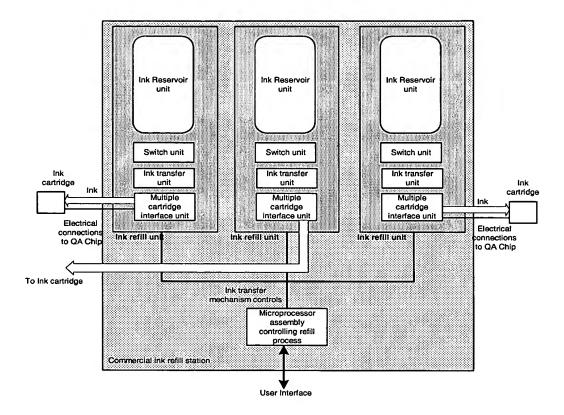


FIG. 360

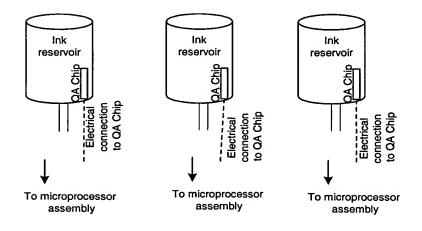


FIG. 361

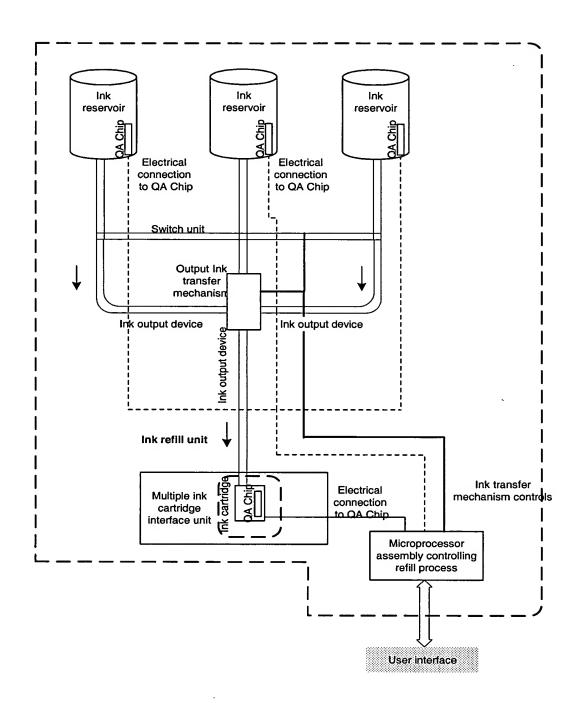


FIG. 362

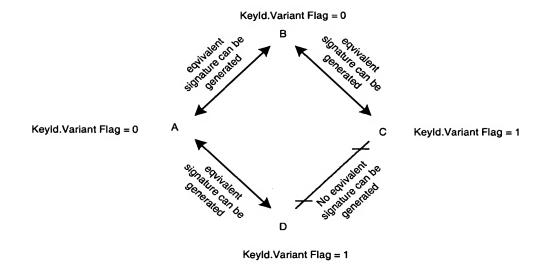


FIG. 363

3	<u> </u>	16 4	3 ()
	Type (15 bits)	Permissions (13 bits)	Size and Position (4 bits)	

FIG. 364

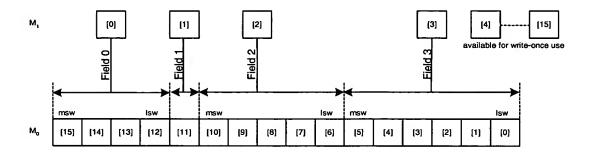


FIG. 365

1	6	14	13	12	11_	4
	KeyNum (3 bits)		NonAuthRW (1 bit)	AuthRW (1 bit)	KeyPerms (8 bits)	

FIG. 366

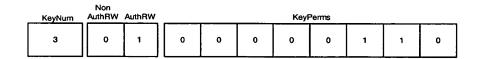


FIG. 367

_	KeyNum	Non AuthRW	AuthRW				Keyl	erms		_		
	3	1	1	0	0	0	0	1	1	1	0	

FIG. 368

31	1	7 16 14	13	12	11	4 3 0
	Type (15 bits)	KeyNum (3 bits)	NonAuth RW (1 bit)	AuthRW (1 bit)	KeyPerms (8 bits)	EndPos (4 bits)

FIG. 369

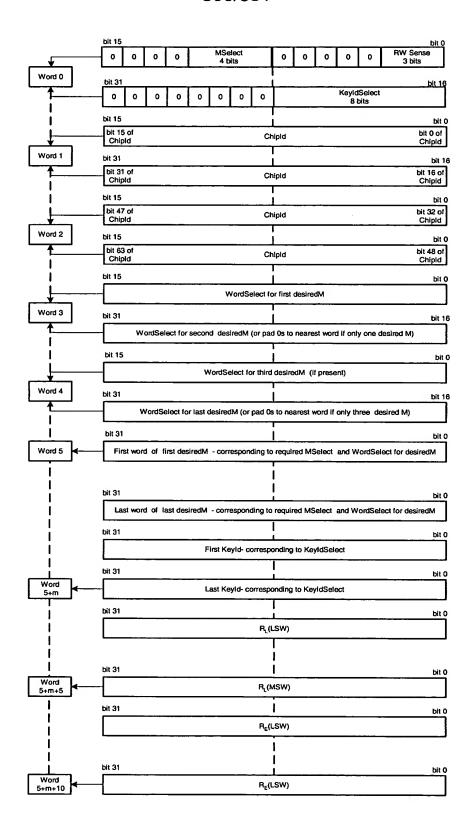


FIG. 370

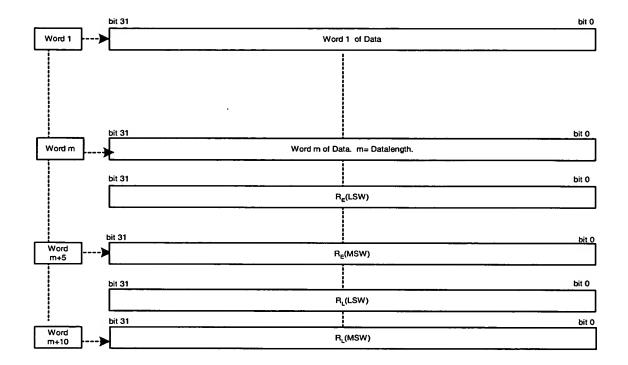


FIG. 371

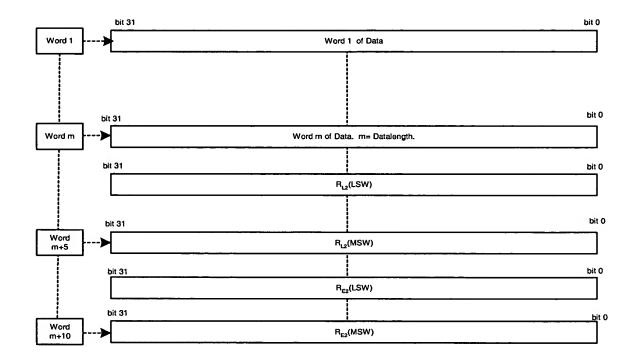


FIG. 372

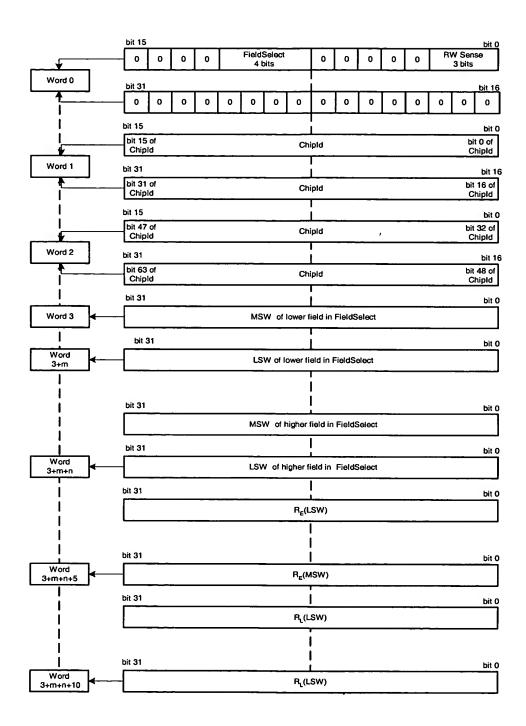


FIG. 373

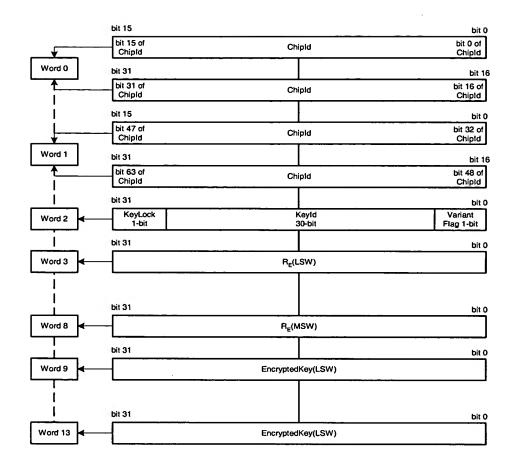


FIG. 374

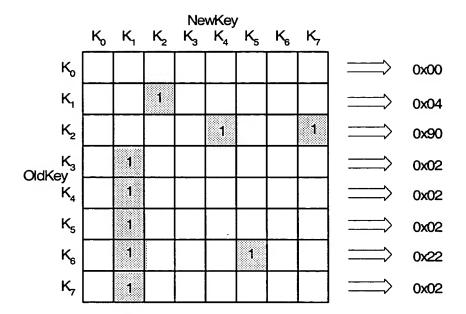


FIG. 375

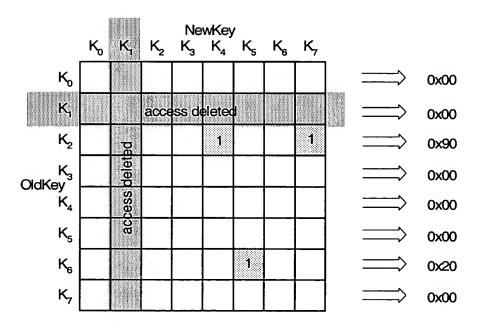


FIG. 376

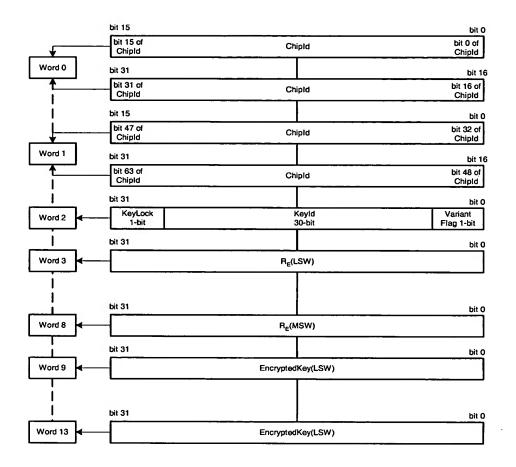


FIG. 377

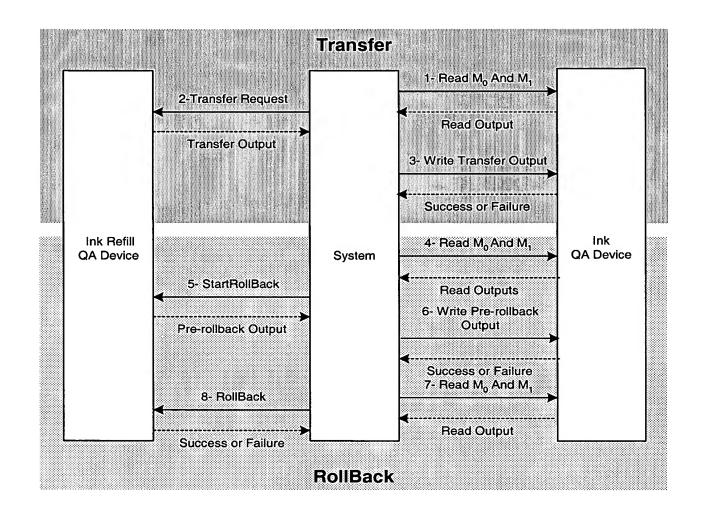


FIG. 378

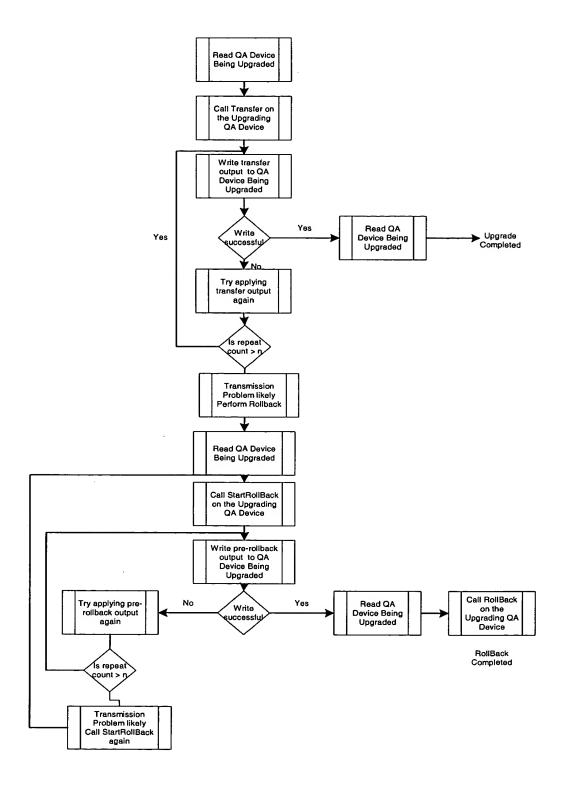


FIG. 379

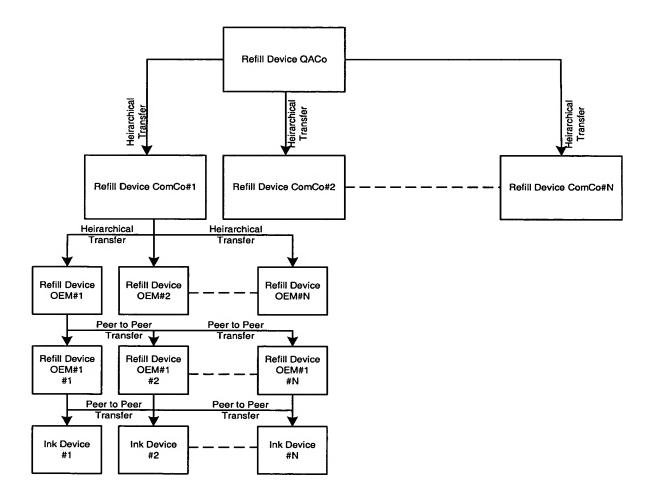


FIG. 380

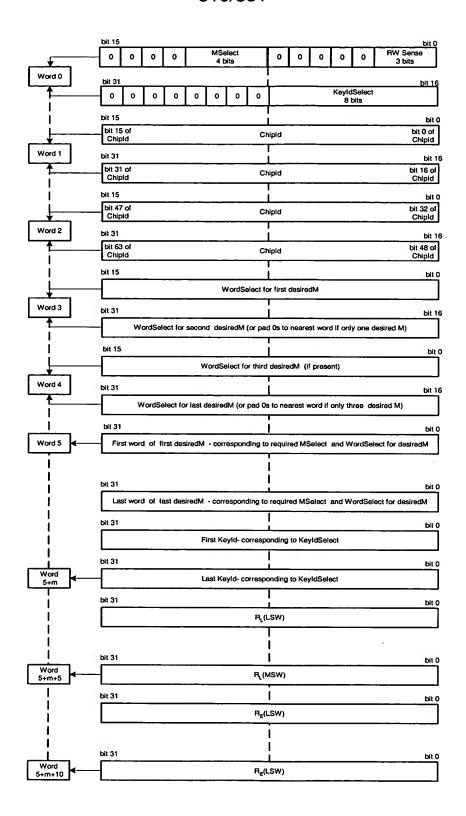


FIG. 381

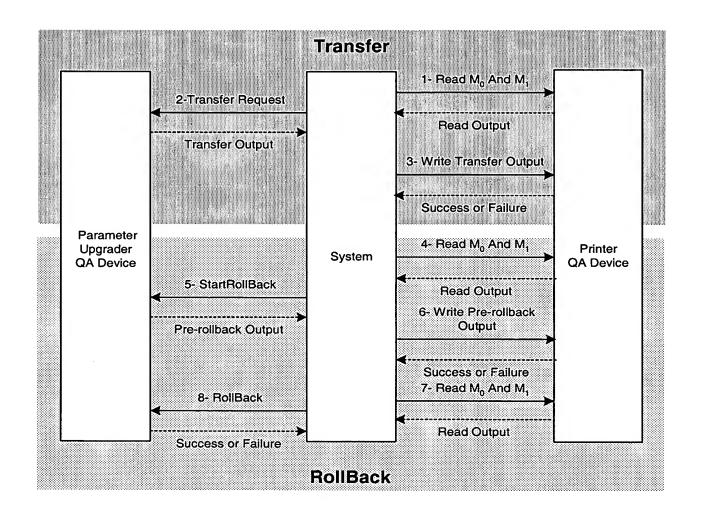


FIG. 382

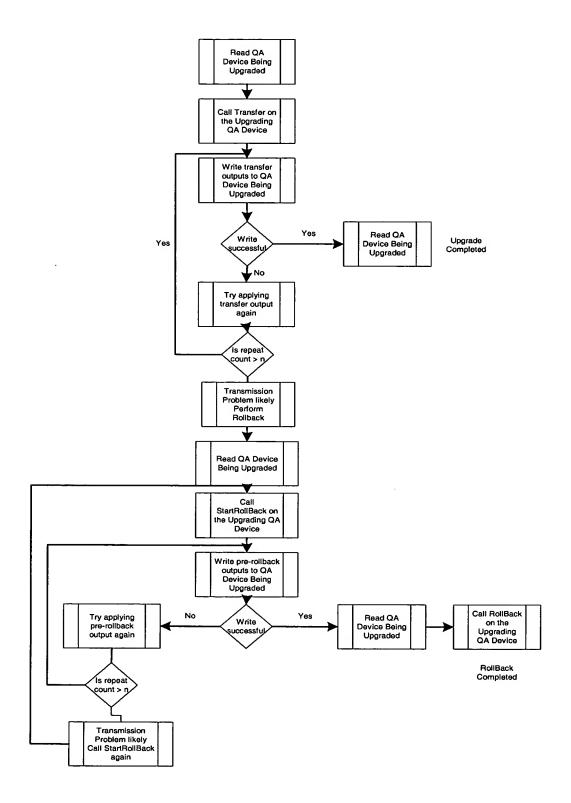


FIG. 383

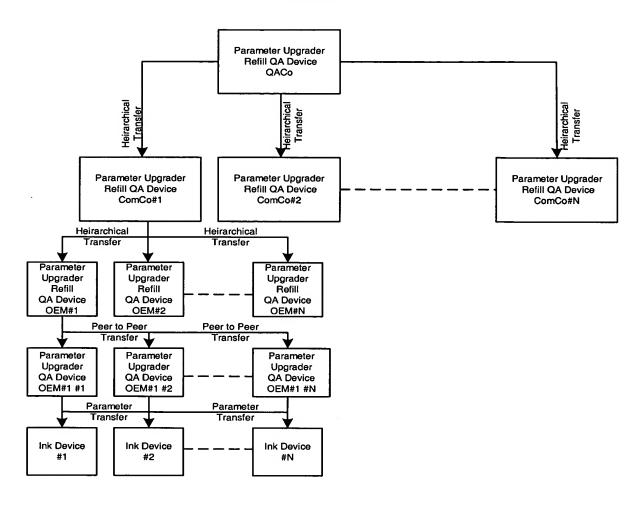


FIG. 384

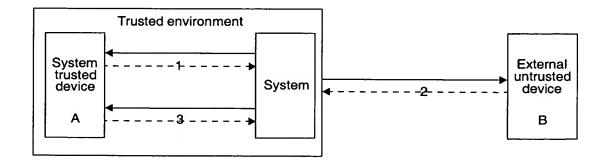


FIG. 385

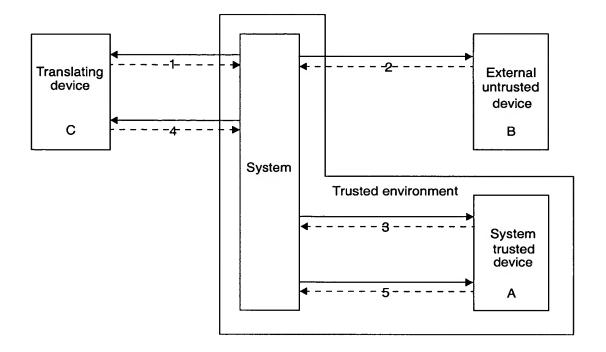


FIG. 386

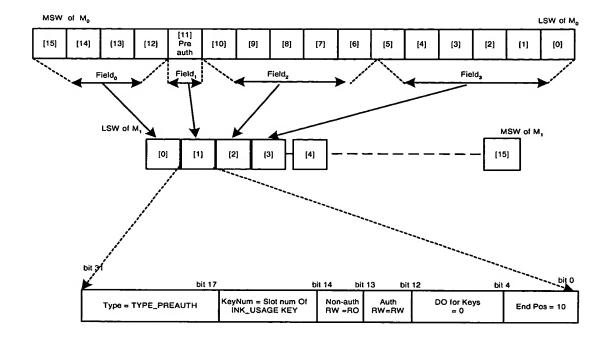


FIG. 387

bit 31	bit (
preauth field select	preauth amount
= 8 bits	= 24 bits

FIG. 388

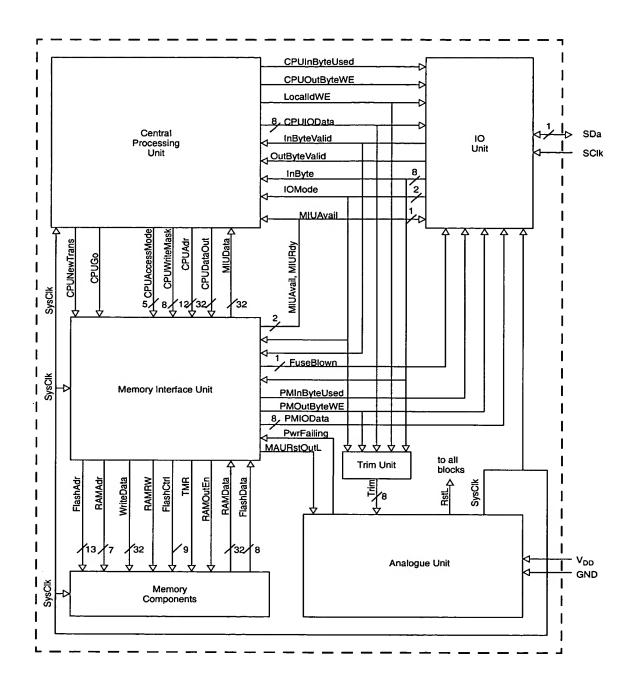


FIG. 389

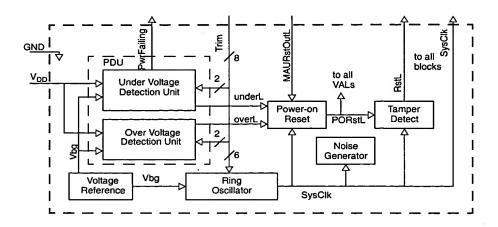


FIG. 390

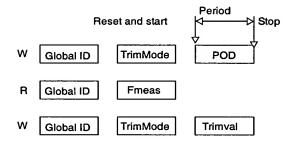


FIG. 391

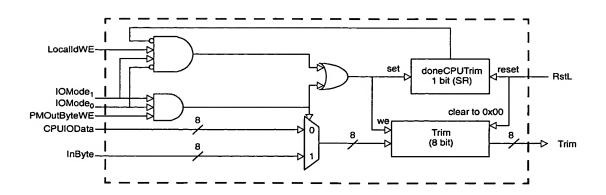


FIG. 392

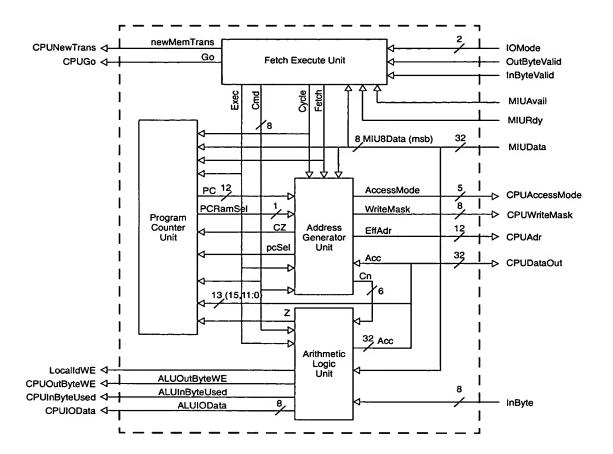


FIG. 393

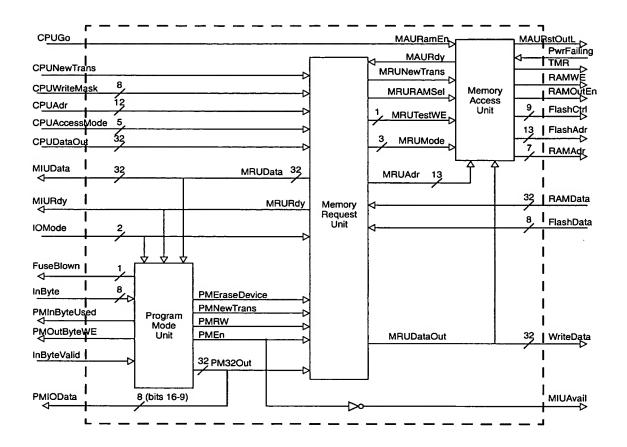


FIG. 394

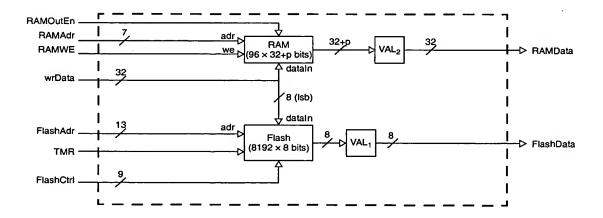


FIG. 395

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PriID6	PriID5	PriID4	PriID3	PriID2	PriID1	PriID0	R/*W 0 = write 1 = read

FIG. 396

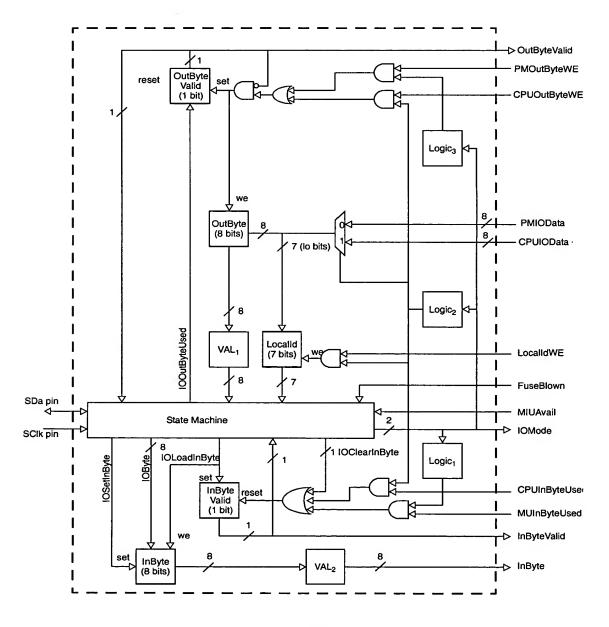


FIG. 397

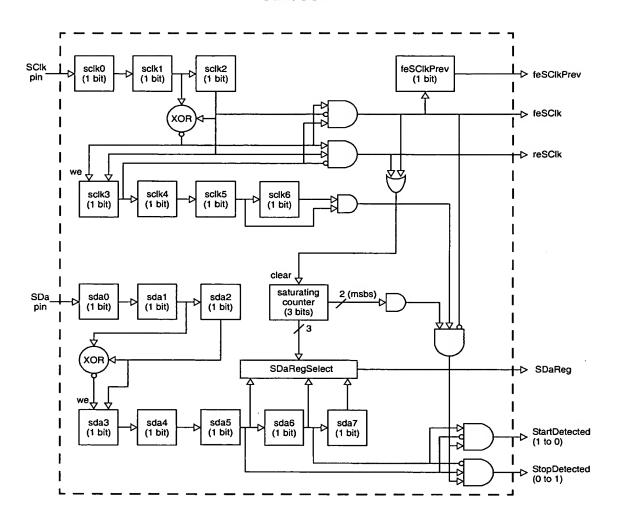


FIG. 398

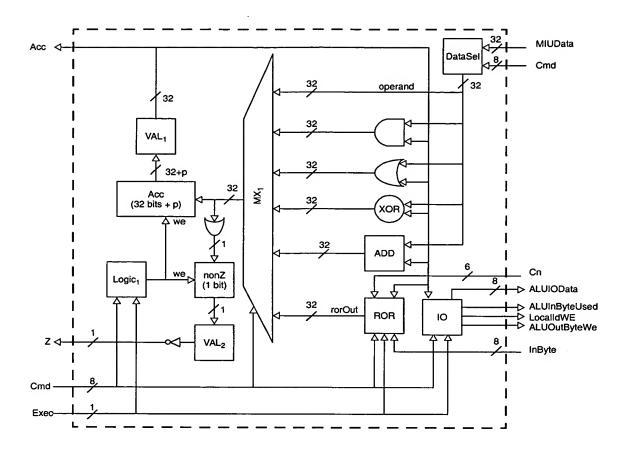


FIG. 399

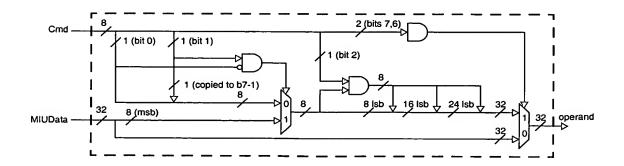


FIG. 400

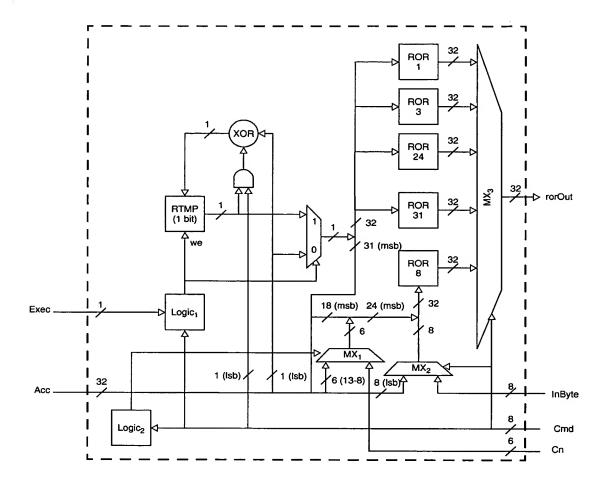


FIG. 401

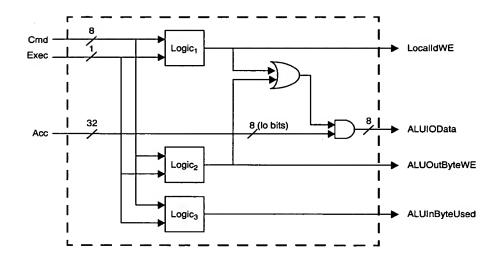


FIG. 402

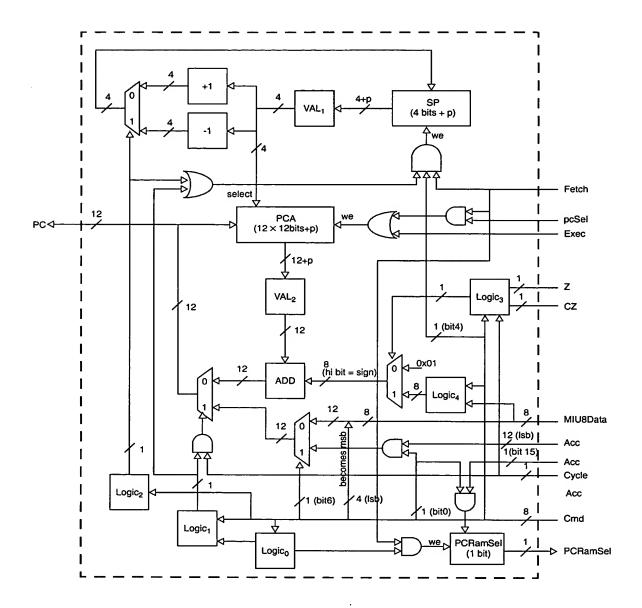


FIG. 403

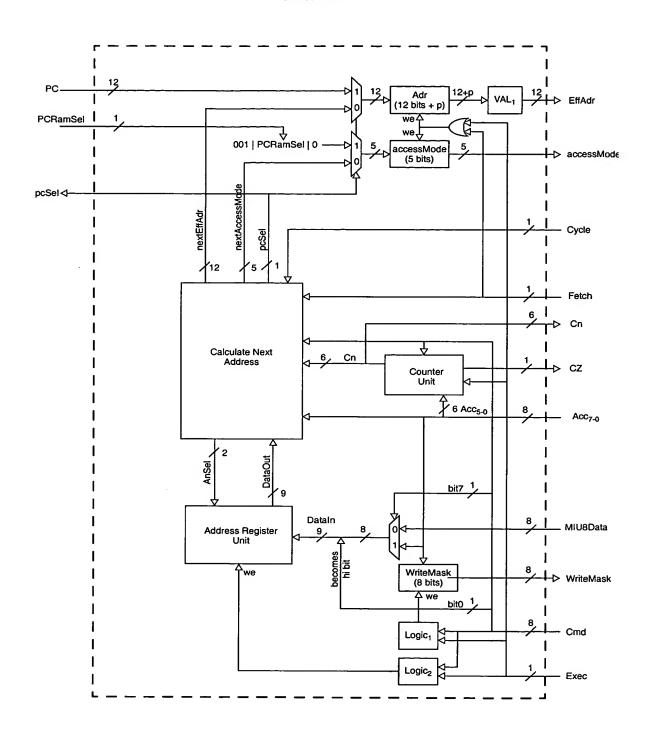


FIG. 404

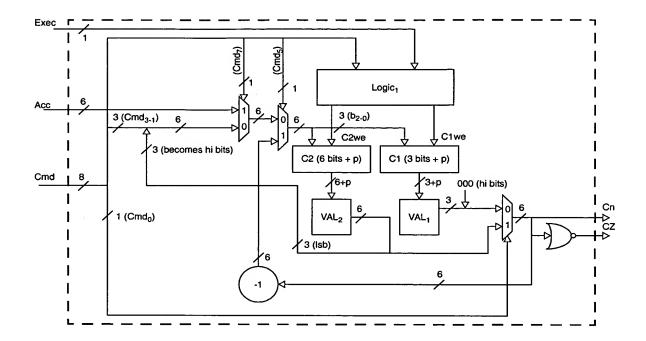


FIG. 405

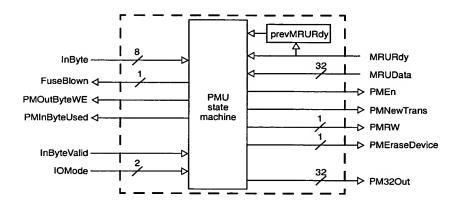


FIG. 406

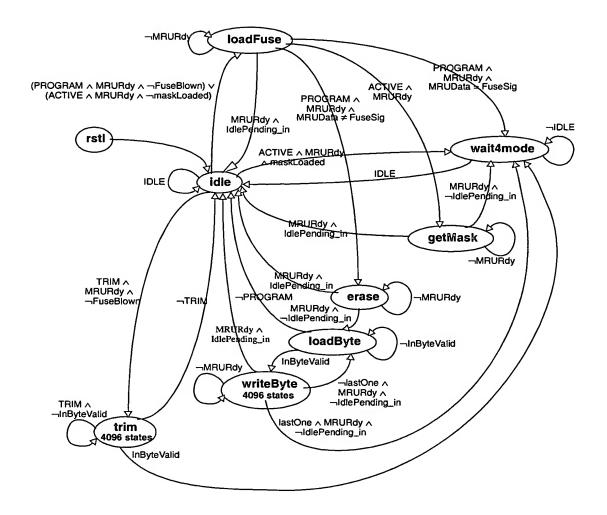


FIG. 407

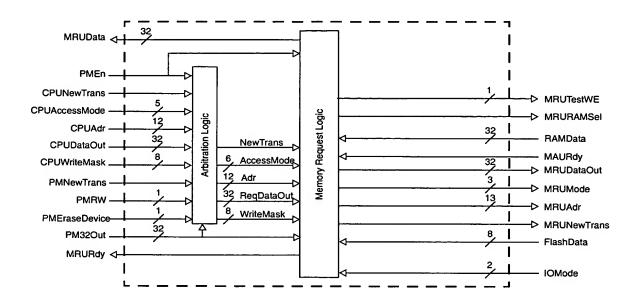


FIG. 408

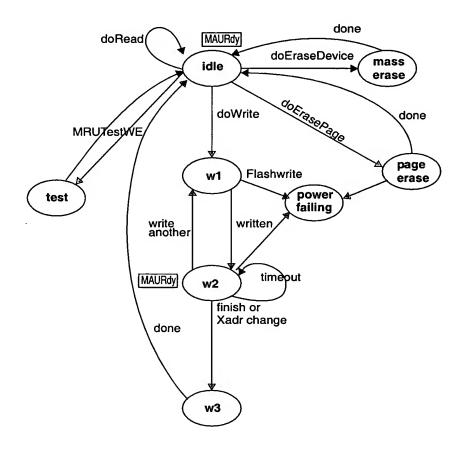


FIG. 409

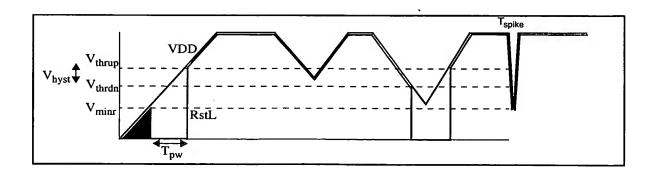


FIG. 410

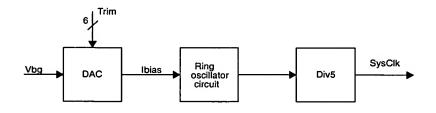


FIG. 411

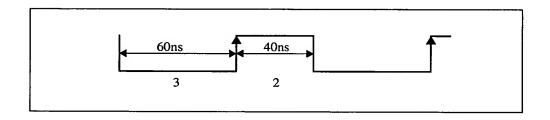


FIG. 412

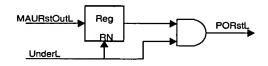


FIG. 413